DATAMOST

$9.95

Kids to Kids

by
Billy Sanders
and
Sam Edge

ON THE COLOR COMPUTER

(for the Radio Shack Color Computer)

TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KID TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KI

Kids to Kids on the Color Computer

TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KID TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KI

O KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

Kids to Kids on the Color Computer

(for the Radio Shack Color Computer)

by

Billy Sanders and Sam Edge

Illustrated by
Martin Cannon

Editor's Introduction
by
William B. Sanders, Ph.D.

DATAMOST™
8943 Fullbright Ave., Chatsworth, CA 91311-2750
(818) 709-1202

DATAMOST

ISBN 0-88190-231-4

TABLE OF CONTENTS

EDITOR'S INTRODUCTION

When we brought our first computer home, our older son first learned how to run and program it. I explained the various parts, commands and got him started programming with a promise of his own computer once he had mastered the essential commands and statements. Like all 9 year olds, his main interest was in arcade games, and slowly he began wondering how he could make his own game. After some extensive work with the computer, he developed a rudimentary adventure game, being led by an interest in making the adventure, rather than the pure joy I always felt in mastering the puzzle of programming.

Shortly after our older son became adept at working and programming his computer, his younger brother wanted to learn as well. I mumbled encouragement and went back to a matter engrossing my attention at the time, and decided in the back of my mind to get around to helping him learn how to use the computer "soon." The next day I noticed that he was already using it, without my "expertise." I asked him if his older brother knew about his using the computer, and he explained that it was his brother who had taught him how to work it. Later on, I found that the older brother could communicate with his younger brother in a way that was clear, concise, and to the point.

Kids have always communicated better with one another than with adults, and this was simply another instance of that phenomenon. Also, kids seem to learn quickly from one another. Adults have ideas about what really interests kids, and sometimes they are right. However, kids just about always know about their mutual interests. And kids *never* want to be told by adults what is good for them. Since the passing on of valued knowledge from kids to kids has been going on since the first cave kid told another about the best trees to climb, why not apply that process to learning about computers? After all, if one kid knows the "neatest" (if they still use that term) thing to do with a computer and how to program it, why not let him or her explain the whole process to another kid?

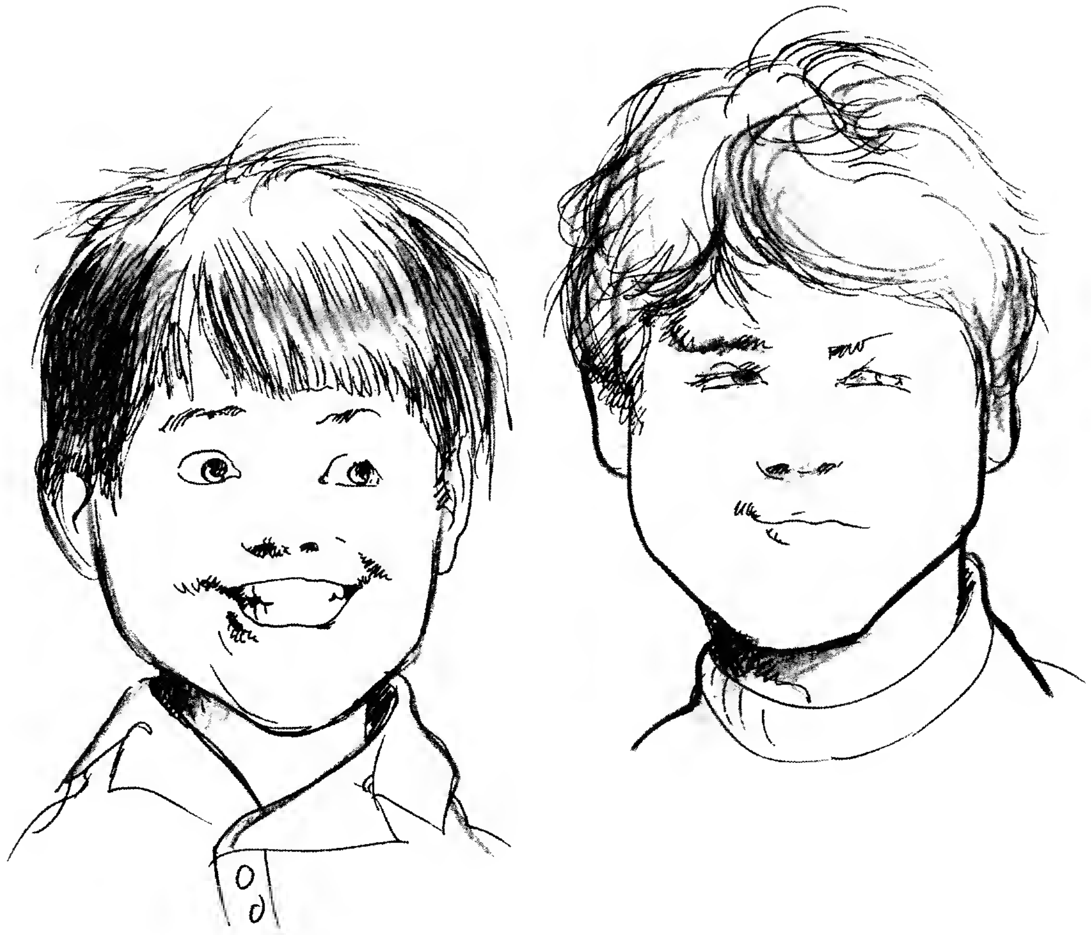
This book was born with the idea that kids can teach other kids better than anyone else. Adult intervention on my part was to give aid and assistance where required, and do all of the grimy work of formatting, correcting, and encouraging. To my surprise and delight, I was not overburdened with editorial tasks, as I had feared. The two young authors dove into their project with enthusiasm and ideas that seemed bottomless. Their energy level was always high, and discussions and phone calls concerning their opus occurred at all times of the day and night.

The level of the book is introductory, but the authors did include some more advanced concepts near the end. This was done so that they could include some games and other programs they felt kids would enjoy. Otherwise, most of the material is designed to help kids get started using and programming their Color Computer.

The two authors, Billy Sanders and Sam Edge are 11 and 17 years old, respectively. They wanted me to include several people who helped them. First, Billy wants to thank Eric Goez for originally suggesting he create the book, Phil Williams of Kraft Systems Company for giving him a Kraft Joystick to help develop programs using the joystick, his brother David, and his parents. Sam would like to thank Don Carothers, Kammie Williams for her support, and the guys from Practical Computing for their general assistance. Sam is also very grateful to his mother for all of her understanding support. Finally, as editor, I would like to thank Billy and Sam for doing such an excellent job.

William B. Sanders, Ph.D.
Series Editor

O KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KID TO



KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KID TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO K

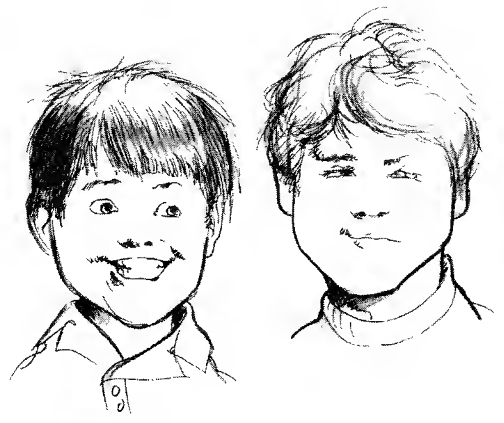
O KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

1
GETTING SET

Hi, we're Billy Sanders and Sam Edge. We are kids like you (11 and 17 years old). That's why this book is called KIDS TO KIDS ON THE COLOR COMPUTER. It's a book for kids from two kids' points of view. We hope this will help you understand how to use your Color Computer. Let's get started by setting up your computer and TV set.

First, you hook up the TV set to the computer. You do that by taking a little box, called a switch box, out of the box you got your computer in. Take the antenna piece from the switch box and hook it up in back of your TV set where the little screws are, and where it says VHF.

There should be a long gray cord in the box your computer came in. Hook one end up to the switch box and the other end up to the computer where it says TO TV. On the back of the computer is a switch for Channel 3 or Channel 4. In some towns, Channel 3 is best, and in others, Channel 4 is best. Turn the switch to Channel 3 or Channel 4 and turn your TV channel selector to Channel 3 or Channel 4.



Now that you have hooked up the TV to the computer, plug the computer into the wall. You do that by taking a cord from the computer and plugging it into the nearest wall outlet.

Next, to turn it on you simply press a little button on the left side of the computer. When you have done that, on the top of the screen it should say something like.

```
COLOR BASIC 1.Ø
COPYRIGHT (©) 1981 BY TANDY
UNDER LICENSE FROM MICROSOFT
```

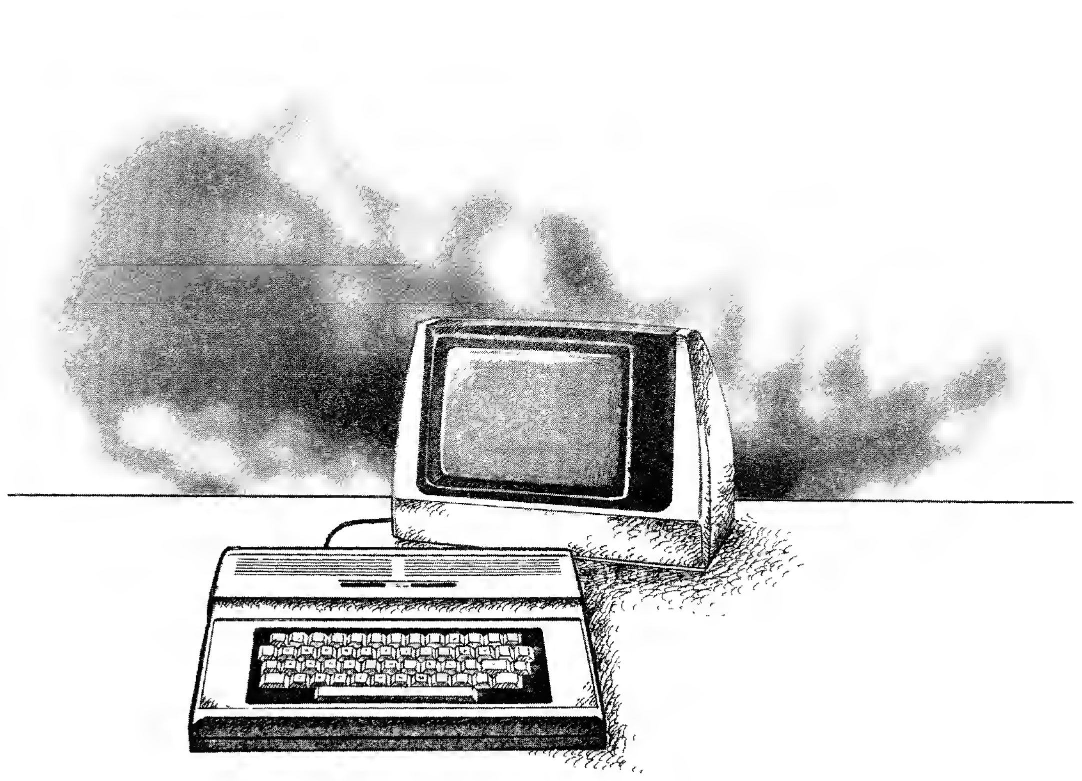
The exact message will depend on when you got your computer and whether or not you have Extended BASIC installed. When you are done with that, play with it for a while. Press some different keys to see what happens.

Some of the keys you have are regular keys and some are irregular. You may know the keys from A-Z pretty well. Also, you know the number keys and their shift statements like !“#$%&’( )*:=–@+;?/>.<,. But what about the up, down, left, and right arrows, and the ENTER, CLEAR and BREAK keys? Here is what they do. The up arrow key makes an arrow like this ↑ . Press shift and the up arrow and it will make a right arrow key like this: → . Press shift down arrow key and it will make a left bracket, and the shift right arrow key makes a right bracket. Press the left arrow key and it will delete the last word you typed and/or it will go back one space. Press shift zero and it will switch back and forth between lower case and upper. With lower case, instead of having black on green it will be green on black. *(Note: It will show lower case only when you print it out on a printer.)* Press shift @ and the computer will pause and then continue when you press any other key. Press shift left arrow key and it will delete the current line. Press the CLEAR key and it will clear the whole screen; if you are working on a program and you are not done with it, that line will be deleted and won't be seen again unless you write it again.

Press the ENTER key and that will end a program line. Press the BREAK key and that will interrupt the line that the program is on. For instance, if you are running a program and press the BREAK key, it will say BREAK IN 2Ø or some other line number. Write in this program to see how it works.

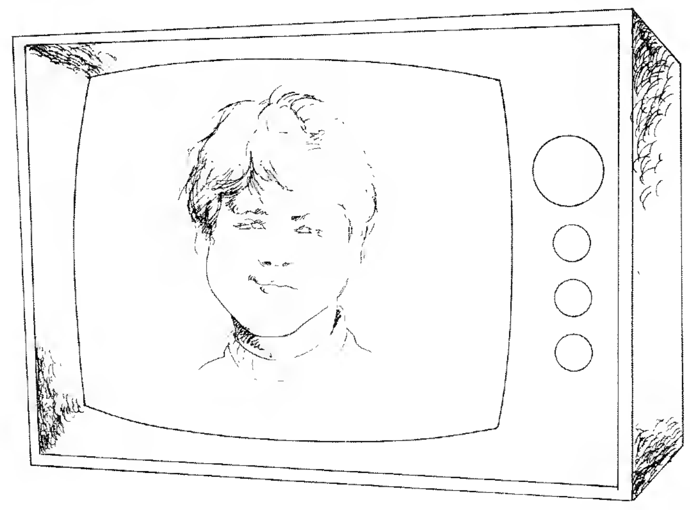
```
1Ø CLS <ENTER>
2Ø PRINT "COMPUTER" <ENTER>
3Ø GOTO 2Ø <ENTER>
RUN <ENTER>
<PRESS BREAK KEY>
```

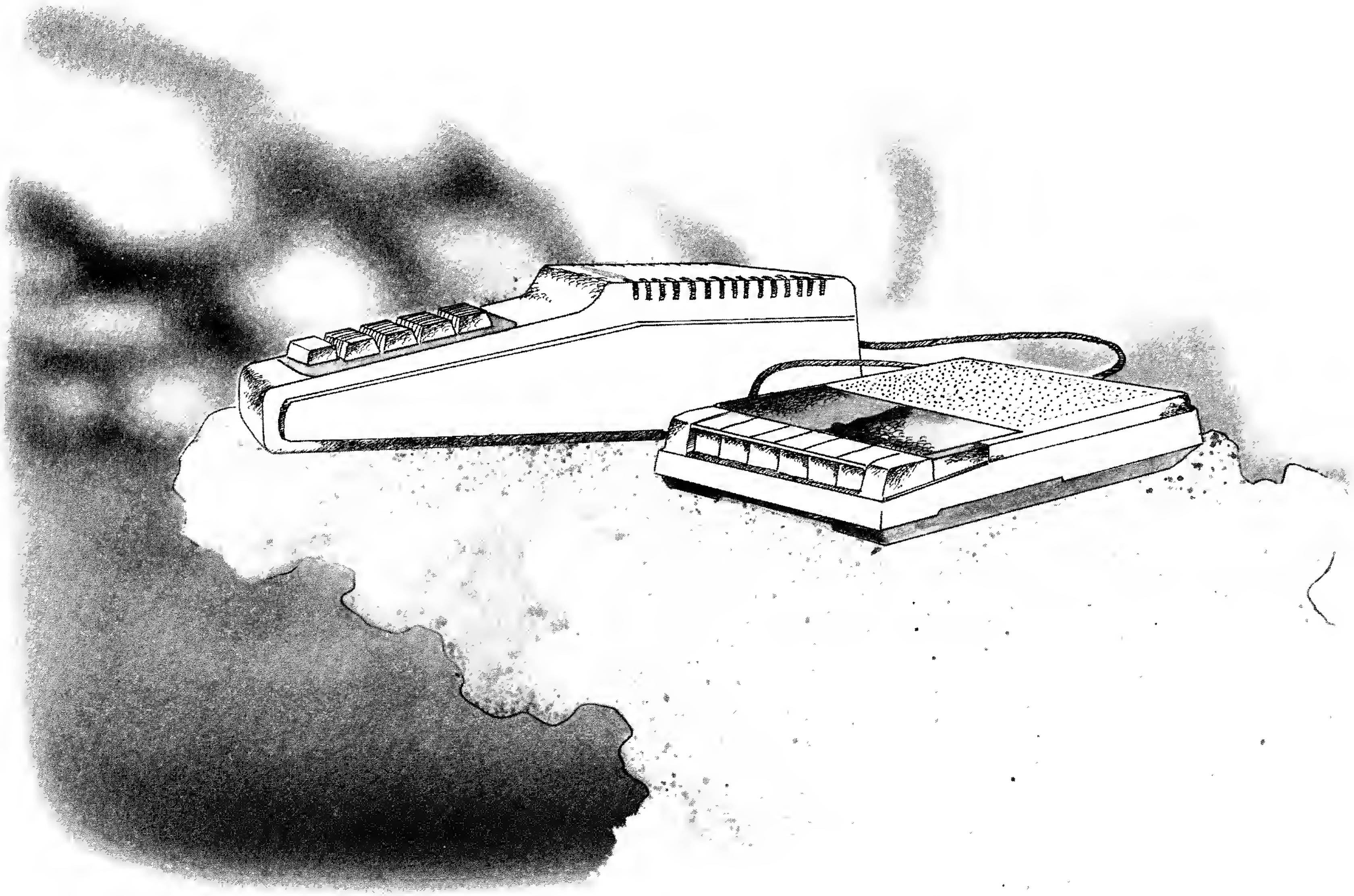
Now you should have the computer hooked up to the TV and you should know what all the keys mean. In the next chapter you will learn how to use the cassette recorder.



2
HOW TO USE THE CASSETTE RECORDER AND THE DISK DRIVE

In this chapter you will be learning how to use the cassette recorder first and the disk drive second. If you want to use your cassette recorder, you have to hook it up first. Turn off your computer and TV. In fact, you should unplug both to be certain. The cord for the recorder has a black box on the end of it that plugs into the wall. Take the other end and hook that up to the cassette recorder where it says 12ØV. *(Note: We would strongly recommend that you buy the Radio Shack TRS-8Ø Computer Cassette Recorder since not every cassette recorder will work with your computer.)* Next, find the cord with three other cords attached to it. Plug the big end into the computer where it says CASSETTE. Then on the other end of the cord you will find three other connectors. Plug the black one into the spot where it says EAR. Then, with the two gray cords, plug the left one into the spot where it says MIC and plug the other one into the spot where it says AUX. Now plug in the computer, TV, and recorder and turn them all on.





Now that you have hooked it up and turned on, here's how to use it. (You will need a tape with programs on it. If you have no such tapes, get a blank tape and skip to the next section that explains how to save a program to tape.) The first word we will introduce to you is CLOAD. CLOAD means Cassette LOAD. How it works is simple. Put your tape with some programs on it in your recorder. You type CLOAD, quotes, the program's name, and then ending quotes.

CLOAD "PROGRAM" <-Use the actual name of the program.

Now press PLAY on your cassette recorder. Then just press ENTER. *(Note: Make sure that you have fast forwarded the tape or rewound it to the program you want. If this is not done, the computer will go searching for the program until it has found it or gives up. You have to watch the cassette counter. Those are the numbers on the cassette recorder that tell where the tape is. By writing down the numbers of the cassette counter, you can tell where your program begins. Then you don't have to wait all day for the recorder to find your program.)* The recorder's red light should go on, and it will start moving forward. When it says OK on the top of the screen, type RUN and press ENTER; you will have your program working. If you get an error when you type CLOAD, try typing CLOADM. You use it the same way you use CLOAD, but when you're finished with your program type EXEC.

One very helpful command when you are using the cassette recorder is SKIPF"X". SKIPF"X" searches through the whole tape until it gets to the end. The computer will print every program that is on that tape at the top of the screen. To locate a program using SKIPF"X", watch the program counter. When the name of a program appears on the screen, look at the cassette counter and write it down along with the program name. Then you can keep track of the programs' locations if you forgot to do it when you first wrote the program.

Sometimes the computer will give you an IO error because the program is not found or the cassette is not hooked up right. If that happens, make sure your cassette recorder is connected where it should be.

SAVING PROGRAMS ON TAPE

So far you have learned how to load programs and search for them. How about learning how to save programs? Saving programs is simple. When you are finished with a program and you want to save it, all you do is type CSAVE, quotes, (") what your program's name is, and then ending quotes ("). See how easy it is! Enter the example program and use it for practice. At the end of every line press the ENTER key. (If this is your only program, go back to the section on CLOAD and practice loading the program.)

```
10 CLS
20 PRINT "SAVE ME"
30 END

CSAVE "THIS PROGRAM" <-Use your program's real
name.
```

'O KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KID

SAVED PROGRAMS

continue to work for you

TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KI

DISK DRIVE

If you do not have a disk drive, you can skip to the next section. If you do not have extended BASIC in your computer, your disk drive will not work; so if you have a disk system, go have Radio Shack put the EXTENDED BASIC chip in for you. It costs about $1ØØ.

To use your Disk Drive, you *first* have to hook it up. Be sure to first unplug your computer and disk drive. You simply take the disk cartridge that looks like a flat black box and put it into the slot on the right side of the computer. Connect the flat ribbon cable from the disk drive to the cartridge. Last, plug in the power cord to a wall outlet. Once you have finished that, find the little switch in the back of the disk drive turn and it on. Now that you have your disk drive all hooked up, you will learn how to use the it. As a reminder, before you do that you must make sure you have extended BASIC. You will learn more about extended BASIC in Chapter 15.

Once you have connected your drive, you have to format a disk for it to work. You format a disk by typing DSKINIØ. When you are finished typing that, press ENTER. The red light on your drive will go on and it will start running. After the light on the drive goes out, your disk is formatted. Now type in DIR. The contents of your disk will appear. Since there are no programs SAVEd to your disk right after you format it, the screen will show nothing. But, for example, let's say there are some programs on that disk. This is what your screen would show:

```
OK
DIR

MAIL LIST  BIN 2 Ø 7
ZAP        BAS 4 Ø 2
WSS GAME   BAS 3 5 5
SAMS GAME BIN 3 Ø 3

OK
```

You might be wondering what the BIN and BAS mean. Well, their meanings are simple. BAS means BASIC. When you want to LOAD a program that says BAS beside it, you simply type LOAD, quotes, the name of the program, and then ending quotes. When the red light on the disk drive goes off, and it says OK below where you typed LOAD, type RUN, and you will have your program.

BIN means BINARY or machine language. If it says BIN next to the program that you want, type what you would type if it said BAS next to the program, but instead of typing LOAD, type LOADM. When the red light goes off and it says OK below the spot where you typed LOAD, type EXEC and you will have your program. EXEC is for EXECUTE.

Now that you have learned how to LOAD programs, how about learning how to save them? Saving is simple. All you type is SAVE, quotes, your program's name, and then ending quotes.

```
SAVE "name of program"
```

The red light on your disk drive should go on, and when it goes off, your program will be SAVEd on that disk. Whenever you type DIR, the name of the program that you saved will show up on the screen. By the way, DIR stands for DIRectory.

In this chapter you have learned how to SAVE and LOAD on both the cassette recorder and the disk drive. These commands deal with I/O (Input/Ouput) to external devices. In the next chapter you will be learning how to use PRINT and math statements.

O KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KID TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KI

TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO K

TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KID

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

3
HOW TO USE PRINT AND MATH STATEMENTS

In this chapter you will be learning how to use PRINT statements first with words and then with math statements. PRINT statements are very easy to use; they are also the first and the most important statements in programming. When you are using the PRINT statement you must always have quotes after it if you are not using variables. (You will learn about variables in the next chapter.) Look at the sample below to get a view of what a PRINT statement looks like.

```
10 PRINT "HI MOM ANO DAO"
20 PRINT "HOW IS EVERYTHING GOING?"
30 PRINT "EVERYTHING IS FINE OVER HERE, SON."
```

When you are finished typing the program type RUN, and on your TV screen will show:

```
HI MOM AND OAD
HOW IS EVERYTHING GOING?
EVERYTHING IS FINE OVER HERE, SON.
```

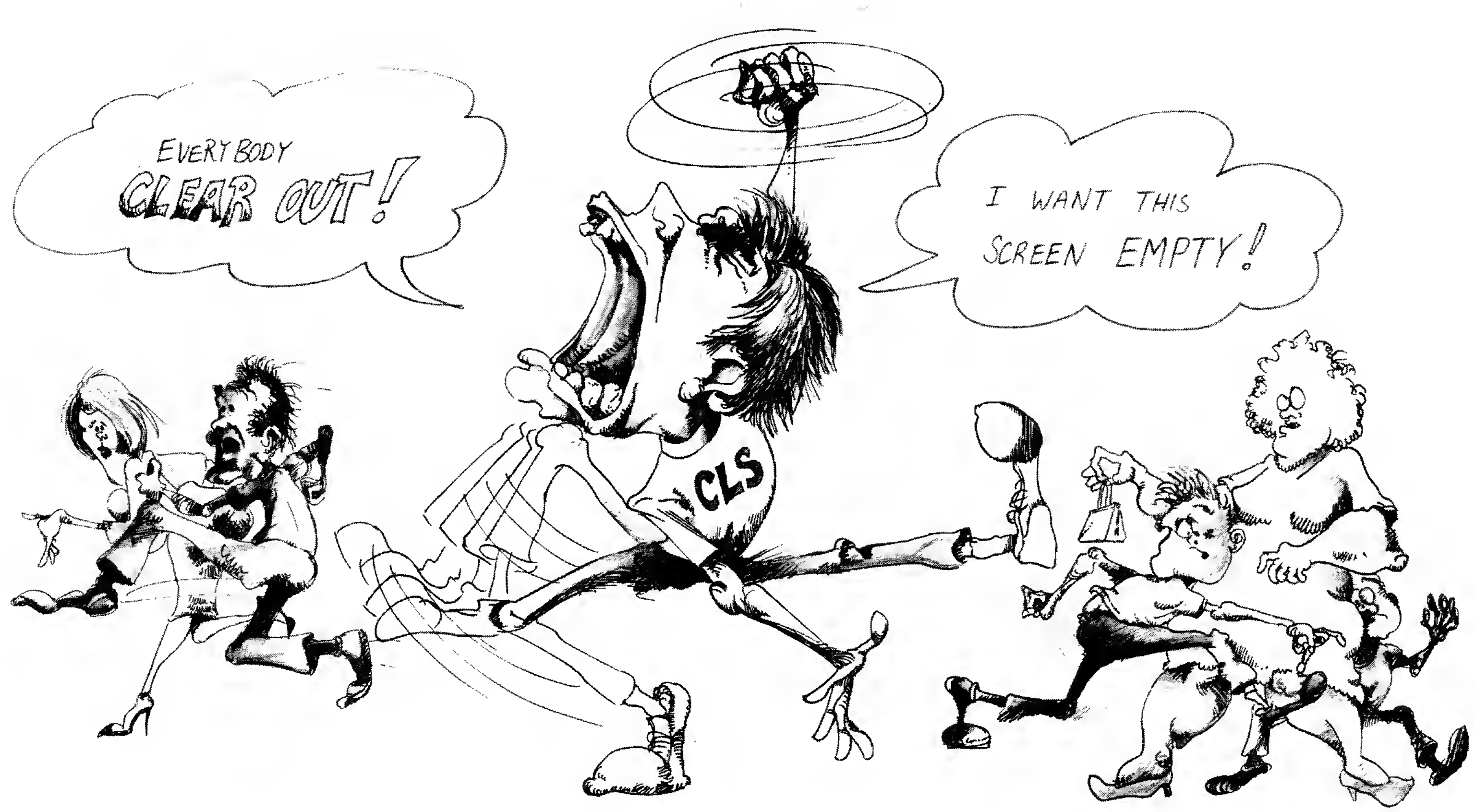
The numbers 1Ø, 2Ø, and 3Ø stand for the line numbers. Every time you want to write a program, you must type line numbers. Line numbers are numbers from Ø to the total number of lines in your program. Usually you start a program with line number 1Ø and number the rest of the program by 1Øs. That's because you might want to add some more statements between the lines. For example, if you number your lines 1, 2, 3 and you want to add something between lines 1 and 2, you would have to start your program all over. But if you number your program 1Ø, 2Ø, 3Ø, as we did above, you could put line 15 between lines 1Ø and 2Ø if you wanted.

Enough with line numbers, let's get back to PRINT statements. A short way to do PRINT is to type the question mark. Type the program we just showed you but type a question mark instead of PRINT. When you type RUN <ENTER>, the question mark <?> should do the same thing as PRINT.

To get used to using the PRINT statement, type in the next program. We will use the question mark this time. Also, we will introduce a new statement, CLS. This will clear the screen for you. We also put in PRINT without PRINTing anything. When RUN, this will put a space in between lines for you.

```
1Ø CLS
2Ø ? "WHAT MONSTER WOULD YOU LIKE TO BE?"
3Ø PRINT
4Ø ? "1. DRAGON"
5Ø ? "2. OGRE"
6Ø ? "3. WEREWOLF"
7Ø PRINT
8Ø ? "YOU GUYS SCARE ME!"
```

TO KIDS TO KI



When you RUN the program, it will look like this:

```
WHAT MONSTER WOULD YOU LIKE TO BE?

1. DRAGON
2. OGRE
3. WEREWOLF

YOU GUYS SCARE ME!
OK
```

Before we go on, we will learn another new statement, LIST. After RUNning your program, type in the word LIST. Your program will then be LISTed to the screen as soon as you press ENTER. (Remember to always press ENTER after an immediate command and after a program line. Words like RUN and LIST are immediate commands.) There's a surprise! Instead of questions marks, there will be PRINT statements where you put the question marks. It will look like this on your screen:

```
1Ø CLS
2Ø PRINT "WHAT MONSTER WOULD YOU
    LIKE TO BE?"
3Ø PRINT
4Ø PRINT "1. DRAGON"
5Ø PRINT "2. OGRE"
6Ø PRINT "3. WEREWOLF"
7Ø PRINT
8Ø PRINT "YOU GUYS SCARE ME!"
```

For practice, try saving the program to tape or disk. We will call the program MONSTER.

CASSETTE
```
CSAVE "MONSTER" <ENTER>
```

DISK
```
SAVE "MONSTER" <ENTER>
```

Now that you have learned how to PRINT letters, how about learning how to do arithmetic? Arithmetic is simple. It is just adding, subtracting, multiplying, and dividing. (It is much more than that, but that's all we're going to show you in this book.) First let's start with adding. Adding is the first thing a kid should know in arithmetic. Look at the example below to see how to do an adding statement.

```
PRINT 2+3
```

When you press ENTER, your screen will look like this:

```
OK
PRINT 2+3
5
OK
```

The PRINT statement tells the computer to PRINT the sum of 2+3 to the screen. If you put quotes around the 2+3, the computer would not PRINT the answer, but instead it would look like this:

```
OK
PRINT "2+3"
2+3
OK
```

So remember if you put quotes around something the computer will PRINT whatever is inside the quotes, but if there are no quotes around numbers, the computer will PRINT the numbers or their arithmetic results.

Another thing a kid should know is how to subtract. Subtracting on your computer is just like adding except you type a minus sign instead of a plus sign. For instance, look at the sample to see what a subtracting problem might look like.

```
OK
PRINT 23-18
5
OK
```

That was easy, but don't use your computer to get the answers to your homework! You will never learn how to do it in your head or when you do not have your computer with you.

Multiplying works the same way as addition and subtraction. Try typing this on your screen.

```
OK
PRINT 6*7
42
OK
```

When you want to multiply with your computer, use a little star called an asterisk for the multiplication sign. You wouldn't use the X because to the computer it just means the letter X. Division is the same as all the others except it uses the slash mark that looks like this /. You might type this to figure a division problem.

```
OK
PRINT 4/2
2
OK
```

For some division problems with fractions, your computer will print up to eight decimal points. When you get to junior high or high school, you will need all those decimal points. The next program gives examples of all the different math functions.

```
1Ø CLS
2Ø PRINT "ADD PROBLEM: 28 + 49"
3Ø PRINT "THE ANSWER IS "; 28+49
4Ø PRINT
5Ø PRINT "SUBTRACT PROBLEM: 83 - 49"
6Ø PRINT "THE ANSWER IS "; 83-49
7Ø PRINT
8Ø PRINT "MULTIPLY PROBLEM: 19 * 51"
9Ø PRINT "THE ANSWER IS "; 19*51
1ØØ PRINT
11Ø PRINT "DIVISION PROBLEM: 73 / 14"
12Ø PRINT "THE ANSWER IS "; 73/14
13Ø END
```

Notice that we were able to PRINT both the message THE ANSWER IS *and* the math problem on the same line. The single PRINT statement took care of printing both. We used the semicolon (;) to put the two together on a single line.

In this chapter you learned how to PRINT statements, and do adding, subtracting, multiplying and dividing. You also learned how to clear the screen with CLR, LIST a program, and use line numbers. In the next chapter you will learn how to use variables.

4 HOW TO USE VARIABLES

NUMERIC VARIABLES

In this chapter, we will show you the two kinds of variables. The first one is called a NUMERIC variable. A numeric variable is like a slot where you keep a number. You name the slot with one or two letters and the name works just like a number. You can change the number in the slot and the letters that represent the slot will now hold the new number. Look at the example to see what a numeric variable program might look like.

```
10 CLS
20 B=25
30 C=40
40 PRINT B + C
RUN <ENTER>
```

When you are done typing that, type RUN and below where you typed the program the screen should show 65 (25 +40 equals 65). The B in line 20 stands for 25. That's why it says B=25. On the next line it says C=40; so in the computer's memory that means the variable C stands for 40. The line below that, line 40, means that the computer is to add B and C. It's the same as saying add 25 and 40, which is why 65 was printed below the program.

]

Numeric variables are fun and easy to use. Here are two more programs you can use to see how NUMERIC variables work. Notice how we PRINT both messages and numbers (variables) in the first program. If we use the semicolon (;), whatever we PRINT will be next to the first item we PRINTed. In the second program, before you RUN it, see if you can guess what the computer will PRINT to the screen. We also introduce a new statement, END. This tells the computer that it should stop. You do not need the END statement at the very end of a program, but in some cases we will look at later, you do need it.

```
10 CLS
20 A = 10
30 A1 = A * A
40 B = 20
50 B1 = B * B
60 PRINT "A = ";A
70 PRINT "A1 = ";A1
80 PRINT "B = ";B
90 PRINT "B1 = ";B1
100 PRINT "A1 TIMES B1 =";A1 * B1
110 END
RUN <ENTER>
```

```
10 CLS
20 X = 50
30 Y = 60
40 Z = 70
50 A = X + Y
60 B = X + Z
```

```
70 C = Y + Z
80 PRINT A
90 PRINT B
100 PRINT C
110 END
RUN <ENTER>
```

Write some other programs of your own that use numeric variables.

STRING VARIABLES

The other kind of variable we are going to show you in this chapter is called a STRING variable. A string variable is like a numeric variable except it has a dollar sign on the end of it. String variables store "strings" in slots just as numeric variables store numbers. A string is any message you put in quotation marks. For example, if you say, A$ = "I'M A COMPUTER STAR", the message would be stored in the slot called A$. When you PRINT A$ your computer prints

```
A$ = "I'M A COMPUTER STAR" <ENTER>
PRINT A$ <ENTER>
I'M A COMPUTER STAR
```

All string variables do is take something really big and change it into something small and easy to print. Look at the sample to see what a string variable might look like.

D$ = DOG

```
10 CLS
20 A$ = "<YOUR NAME>"
30 PRINT "HI ";A$
40 END
RUN <ENTER>
```

When you store your name in the slot called A$ in line 20, it will PRINT your name when you PRINT A$ in line 30. We added HI and a semicolon so that your computer would greet you. You can mix string and numeric variables in the same program if you like. Look at the next program.

```
10 CLS
20 AG = <YOUR AGE>
30 N$ = "<YOUR NAME>"
40 PRINT N$; " IS ";AG; "YEARS OLD"
50 END
RUN <ENTER>
```

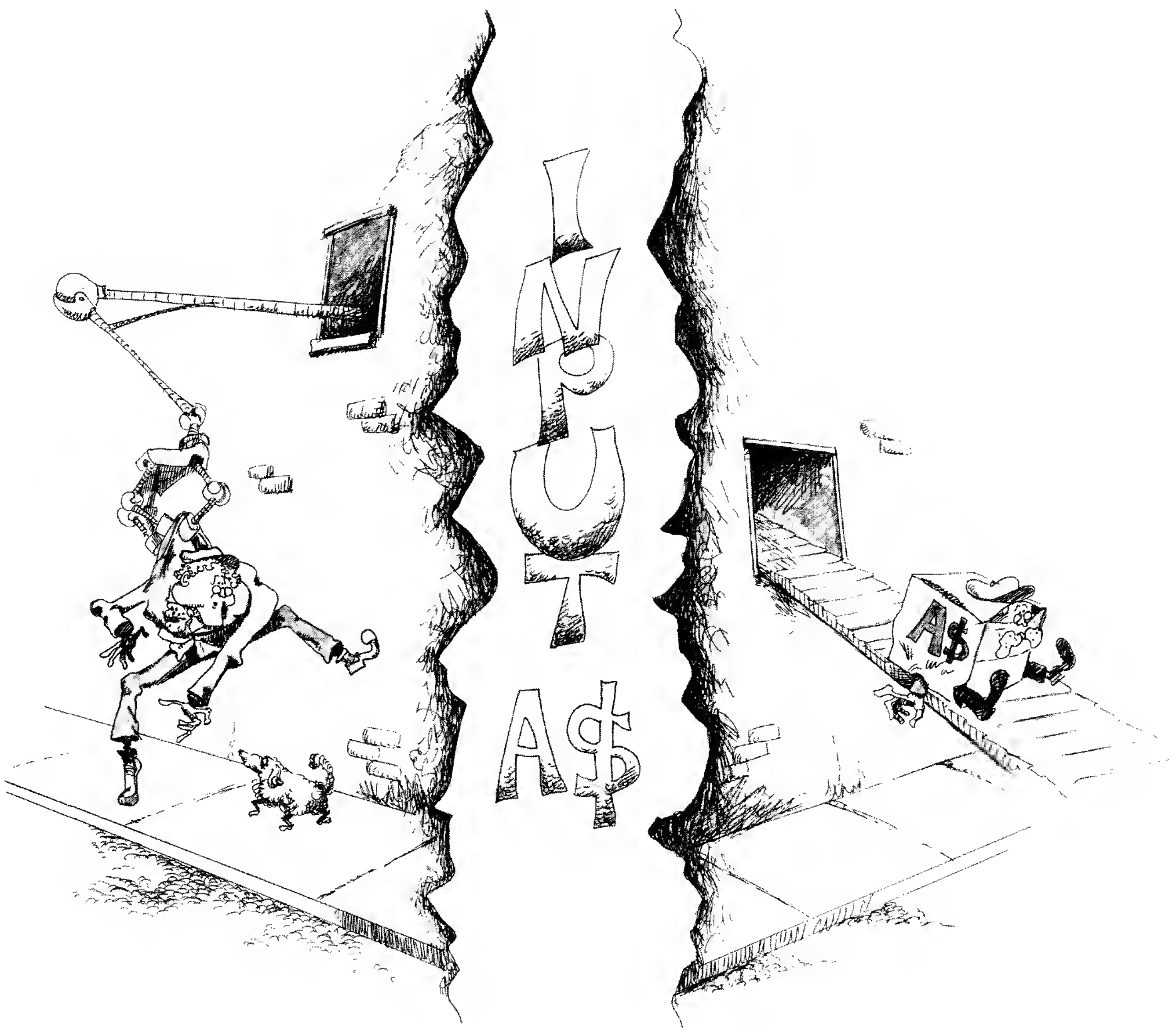
INPUT STATEMENT

The next and the last thing we are going to show you is how to INPUT variables. That again is very simple. Instead of using a variable to equal something such as

```
A$ = "AIRPLANE"
```

we enter the name of the string or the value of the numeric variable after we RUN the program.

```
10 INPUT A$
```


INPUT
A$
A$

Look at the sample to see what a program with an INPUTted variable might look like.

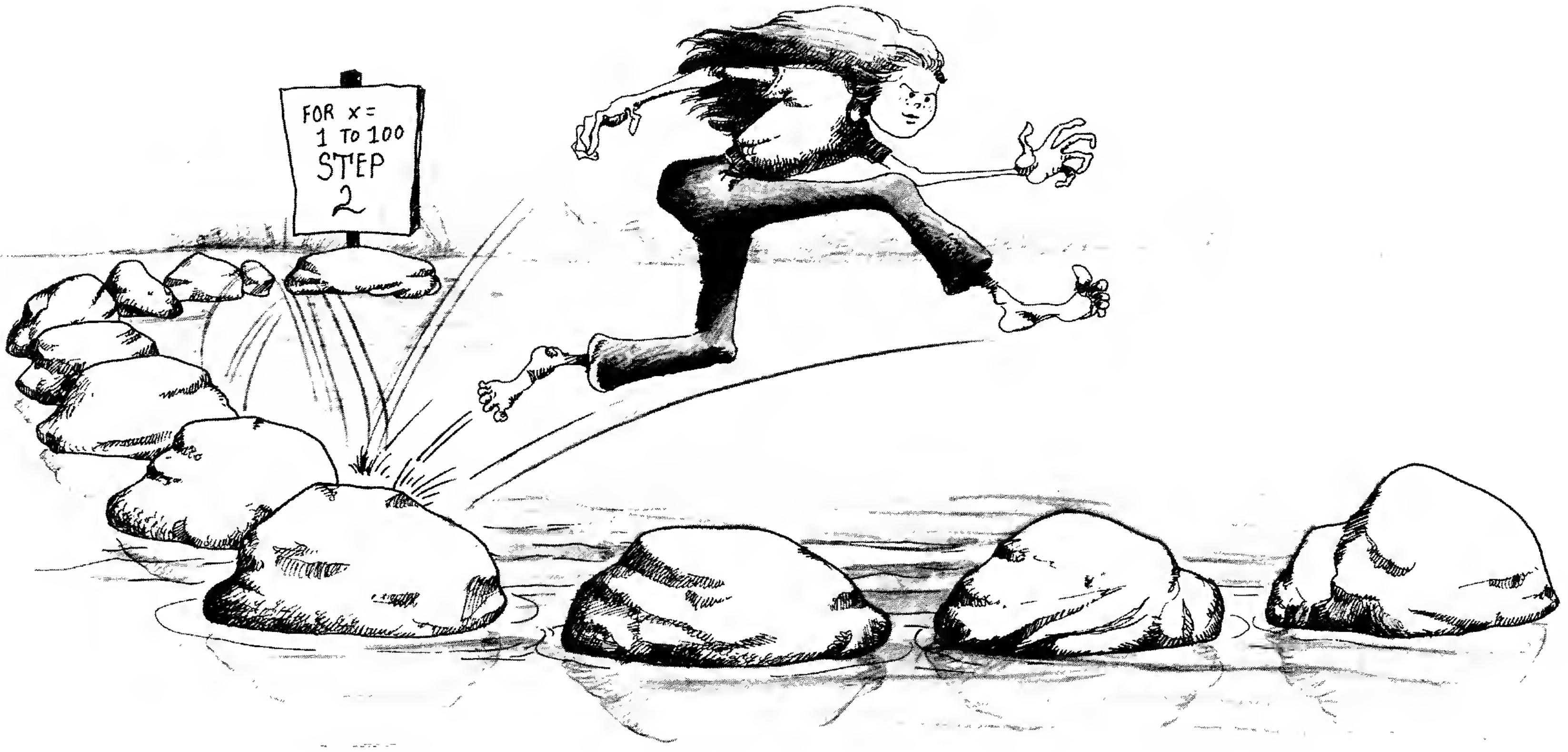
```
1Ø CLS
2Ø PRINT "WHAT IS YOUR NAME?"
3Ø INPUT N$
4Ø PRINT "HOW OLD ARE YOU?"
5Ø INPUT AG
6Ø PRINT "HI"; N$
7Ø PRINT "YOU ARE " ;AG; " YEARS OLD"
8Ø END
```

You saw another program similar to this one earlier, but by using the INPUT statement we were able to enter any name and age we want. INPUT can change the value of the variables when we RUN the program. The next program shows how we INPUT and PRINTed two different strings for the same string variable.

```
1Ø CLS
2Ø PRINT "ENTER WORD NUMBER ONE"
3Ø INPUT A$
4Ø PRINT A$
5Ø PRINT "ENTER WORD NUMBER TWO"
6Ø INPUT A$
7Ø PRINT A$
```

The first time we INPUT A$, it PRINTed the first word. The second time we INPUT A$, it printed the second word. This shows that you can change the contents of a variable while a program is being RUN. Change the program so that you can add two more words. To do that, just add some more lines to the program beginning with line 80.

You can now see how important variables are. They are very useful and very helpful. Well, guess what? This is the end of the chapter. You have learned how to use both kinds of variables — numeric and string variables. You have also learned how to INPUT variables. In the next chapter you will be learning how to use different kinds of loops such as the GOTO statement and the FOR/NEXT loop.


FOR x =
1 TO 100
STEP
2

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO K

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KID TO

O KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

5 USING LOOPS

There are two kinds of loops we will be showing you in this chapter. They are called the FOR/NEXT loop, and the GOTO loop. First, we will show you how to use a FOR/NEXT loop. Look at the example to see what a FOR/NEXT loop looks like. When you are finished looking at it, type it on your computer and RUN it to see what happens.

```
1Ø CLS
2Ø FOR X=1 TO 1Ø
3Ø PRINT X
4Ø NEXT X
5Ø END
```

In the second line, line 2Ø, you will find that it says FOR X=1 TO 1Ø. That means that X equals 1 to 1Ø. The letter X is a type of variable. The value of X begins at 1 and goes up to 1Ø. Each time the program hits the NEXT statement, it loops back to line 2Ø increasing the value of X by 1. That is why it is called a loop. The program does this until the value of X is equal to 1Ø and then it leaves the loop and goes to the line after the statement NEXT. On line 3Ø the program says to PRINT X. So the computer is going to PRINT the value of X. The

FOR
I=1 TO 4
NEXT!
4
3
1

first time through the loop, the value is 1, and then 2, then 3 and so on until it reaches 1Ø. That's why it PRINTs 1-1Ø to the screen. If the loop began with FOR X = 4Ø TO 5Ø it would PRINT 4Ø, 41, 42, etc. until it reached 5Ø. A good use for FOR/NEXT loops is when you have to do the same thing several times. If you have 1Ø names to enter you could write a program that uses a loop from 1 to 1Ø. Look at the next example.

```
1Ø CLS
2Ø FOR X = 1 TO 1Ø
3Ø PRINT "NAME ";X;
4Ø INPUT N$
5Ø NEXT X
6Ø END
```

If you did not have the FOR/NEXT loop, you would have had to do it the hard way. Look at the example of the hard way.

```
1Ø CLS
2Ø REM THIS IS THE HARD WAY
3Ø PRINT "NAME 1 ";
4Ø INPUT N$
5Ø PRINT "NAME 2 ";
6Ø INPUT N$
7Ø PRINT "NAME 3 ";
8Ø INPUT N$
9Ø PRINT "NAME 4 ";
1ØØ INPUT N$
11Ø PRINT "NAME 5 ";
12Ø INPUT N$
13Ø PRINT "NAME 6 ";
```

```
140 INPUT N$
150 PRINT "NAME 7 ";
160 INPUT N$
170 PRINT "NAME 8 ";
180 INPUT N$
190 PRINT "NAME 9 ";
200 INPUT N$
210 PRINT "NAME 10 ";
220 INPUT N$
230 END
```

Using the FOR/NEXT loop it took only six lines to write the program. Using the old way took 22 lines. (We didn't count the REM statement line. All a REM does is to let you put a comment into a program. It doesn't affect the program at all.)

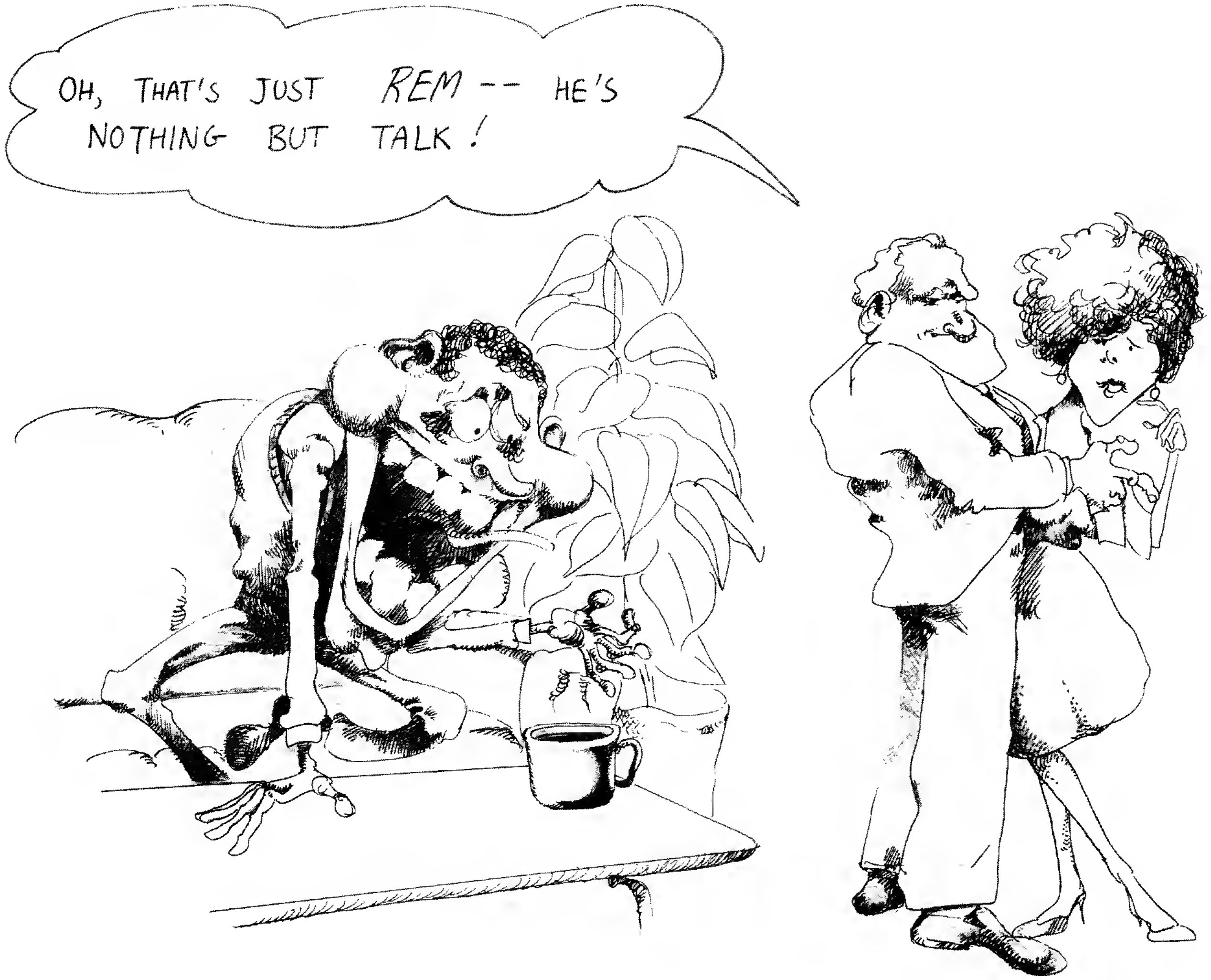
You can also change the value in a FOR/NEXT loop with something other than one. Your computer can count by two's, three's or any other number you choose. To do this you have to use the STEP statement. It looks like this:

```
FOR X = 1 TO 100 STEP 2
```

Instead of counting from 1 to 1ØØ by one's, it does it by two's. Enter the next program to see what happens when you use STEP.

```
10 CLS
20 FOR X = 1 TO 100 STEP 2
30 PRINT X
40 NEXT X
50 END
```

O KIDS TO KID TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO K



Try changing the STEP value to see what happens. If you want to count backwards, use STEP and a minus (-). For instance, you could have

```
FOR X = 1ØØ TO 1 STEP -1
```

Here's a program that will count from 1ØØ to 1.

```
1Ø CLS
2Ø FOR X = 1ØØ TO 1 STEP -1
3Ø PRINT X,
4Ø NEXT X
5Ø END
```

Play with the statements to see what you get. The comma (,) in line 3Ø will print the numbers in two columns. Try changing the comma to a semicolon (;) and a blank (PRINT X) to see the different results on your screen. Now that you know how to use FOR/NEXT loops, let's see how well you can do with the GOTO statement. Look at the next program to see what GOTO might look like in a program.

```
1Ø A$="FLOWER"
2Ø B$="BED"
3Ø PRINT A$;B$
4Ø GOTO 1Ø
```

On lines 1Ø, 2Ø and 3Ø you see string variables that you learned about in the last chapter. Well, what the program says is that A$ = FLOWER and that B$, another string variable, = BED. On line 3Ø it says to PRINT A$ and B$, so on your screen it would print FLOWERBED

THE SIGN SAYS "GOTO", SO WE'D BETTER HEAD THIS WAY!
GOTO 100

once. On line 4Ø the program says GOTO 1Ø so what the computer is going to do is go back to 1Ø and do the program all over again. Your screen will fill up with FLOWERBED. The computer will keep on doing that until you press BREAK, SHIFT @, turn it off, or until it breaks down. This is an example of an endless loop. Endless loops are not used very much since you want your program to end some time. Later on we will see a better use for the GOTO statement when we look at branching and decision making. But since we are talking about endless loops, the next program shows how to make a TV commercial with an endless loop.

```
1Ø REM *****************************
2Ø REM TELEVISION COMMERCIAL
3Ø REM *****************************
4Ø CLS
5Ø PRINT "SEE THE GREAT MOVIE"
6Ø PRINT "THE CAT THAT ATE MIAMI"
7Ø PRINT
8Ø FOR W = 1 TO 1ØØØ
9Ø NEXT W
1ØØ GOTO 5Ø
```

We used both a GOTO loop and a FOR/NEXT loop. The FOR/NEXT loop in lines 8Ø and 9Ø stopped the program for a couple of moments. The GOTO loop went back to line 5Ø and PRINTed the commercial message all over again. Try making your own commercial. For practice, try these next three programs.

```
10 FOR X=1 TO 1000
20 PRINT X
30 NEXT X
40 FOR V=1 TO 100
50 PRINT V,
60 NEXT V
70 PRINT "IF YOU CAN SAY ALL THOSE NUMBERS IN
   ONE MINUTE YOU SHOULD BE ON THAT'S
   INCREDIBLE!"
```

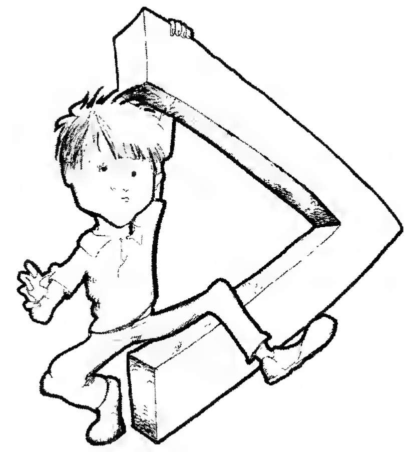
```
10 PRINT "WHAT IS YOUR NAME?"
20 INPUT A$
30 PRINT A$
40 GOTO 30
```

```
10 FOR I= 1 TO 50
20 PRINT I
30 NEXT I
40 GOTO 10
```

In this chapter you have learned how to use two kinds of loops called FOR/NEXT loops and GOTO loops. In the next chapter you will learn how to use branching and subroutines. We will see more of the GOTO statement there.

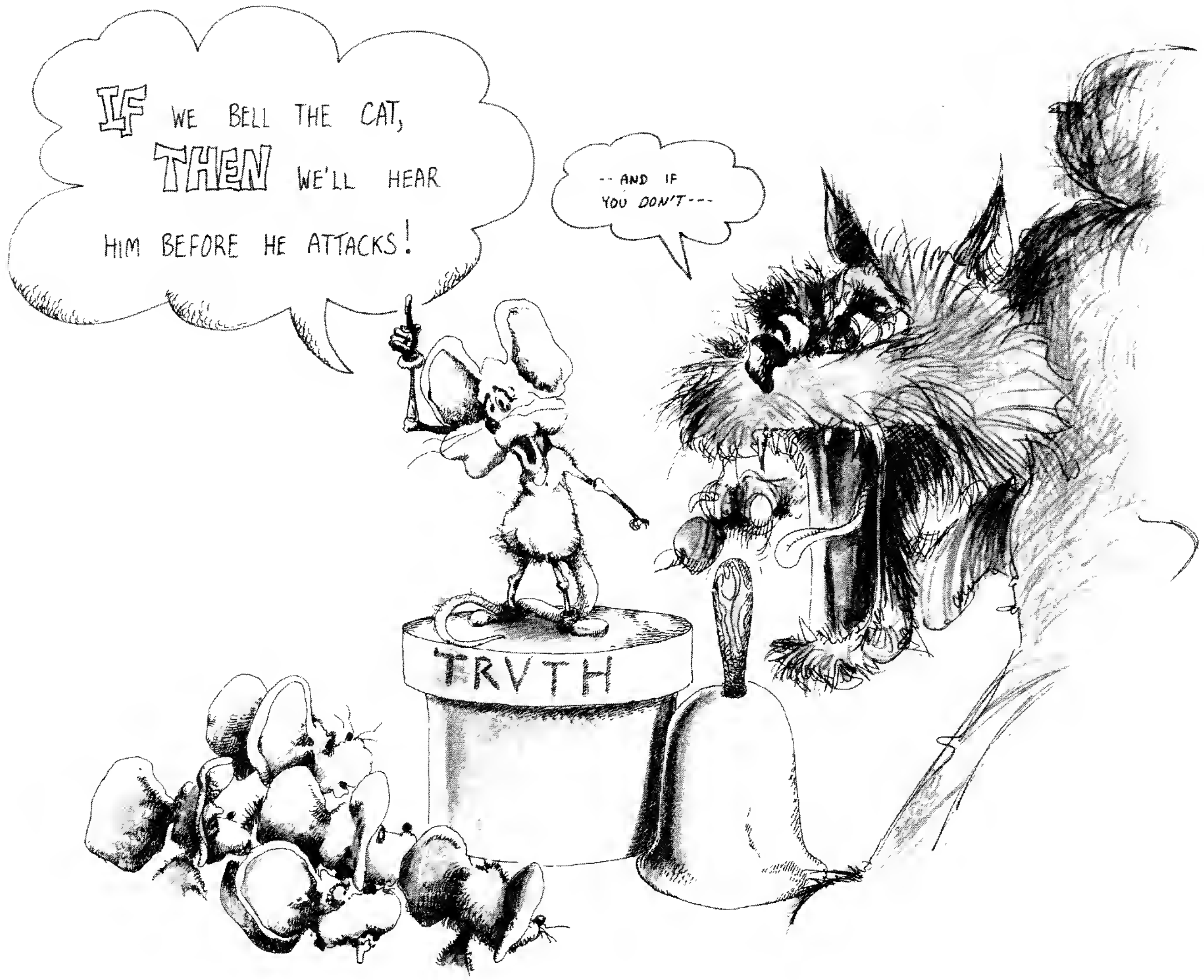
6 DECISION MAKING

In this chapter we will explain and use the IF/THEN statement. It will enable you to do complex branching around your BASIC programs. There are two different types of branches or jumps, the GOTOs and the GOSUBs. These are direct branches. You now ask what the heck is the IF/THEN statement. Type in the following program and let's see.



```
1Ø CLS
2Ø PRINT "ENTER 1 OR 2";
3Ø INPUT N
4Ø IF N = 1 THEN GOTO 1ØØ
5Ø IF N = 2 THEN GOTO 2ØØ
6Ø END
1ØØ PRINT "YOU ENTERED ONE"
11Ø END
2ØØ PRINT "YOU ENTERED TWO"
```

Lines 4Ø and 5Ø used IF/THEN statements. The statements looked to see if the value of N was equal to 1 or 2. IF the value of N was 1 THEN the program branched to line 1ØØ. IF the value of N was 2 THEN the program branched to line 2ØØ. Notice that we used two END statements. If you did not enter a 1 or a 2 then there was no branch and instead the program just ENDed at line 6Ø. We also put an END statement in line 11Ø so


IF WE BELL THE CAT, THEN WE'LL HEAR HIM BEFORE HE ATTACKS!
--AND IF YOU DON'T---
TRVTH

that if you entered 1, the program would not continue to 2ØØ and PRINT the statement YOU ENTERED TWO.

The next program is more complex, but it shows you the power you have using IF/THEN statements. It also introduces the use of multiple statements on a single line. To place multiple statements within a single line, we have to use the colon (:). It works just like putting in another line number, but you can save memory and time by using it instead of a new line.

```
1Ø CLS
2Ø PRINT "WOULO YOU LIKE ME TO TELL YOU A
    JOKE";
3Ø INPUT I$
4Ø IF I$ = "YES" THEN 1ØØ
5Ø IF I$ = "NO" THEN 7Ø
6Ø GOTO 2Ø : REM *** GO ASK AGAIN ***
7Ø PRINT : REM *** SKIP A LINE ***
8Ø PRINT "SORRY TO HEAR IT, IT WAS A GOOO JOKE."
9Ø ENO
1ØØ PRINT "WHY OID THE FOOL ORIVER MAKE
     SEVEN PIT STOPS IN THE INOIANAPOLIS 5ØØ?"
11Ø FOR A = 1 TO 15ØØ : NEXT A :
     REM *** WAIT A FEW SECONDS ***
12Ø PRINT : REM *** SKIP A LINE ***
13Ø PRINT "TWO FOR GAS ANO FIVE
     FOR DIRECTIONS!!!"
14Ø ENO
```

In this program the computer asks you to INPUT whether or not you would like to see the joke. You answered by replying YES or NO. If your INPUT was YES, the computer then branched to line 1ØØ because of the IF I$ = "YES" THEN GOTO 1ØØ. Now, if you typed NO , the computer made the decision not to branch to 1ØØ because your INPUT was not equal to YES. Therefore it saw in line 5Ø that the INPUT was NO and so branched to 7Ø. Line 6Ø is just an error trap for any type of wrong INPUTs, so anything that was INPUTted other than YES or NO would cause the computer to jump back to 2Ø and ask the question again.

The next program is a simple game that will demonstrate some of the logical functions shown below. They are called relationals.

SYMBOL	**MEANING**
=	EQUAL TO
>	GREATER THAN
<	LESS THAN
< >	NOT EQUAL TO
>=	GREATER THAN OR EQUAL TO
<=	LESS THAN OR EQUAL TO

All these functions can be used with the IF/THEN statement to form complex decision making. Type in the following program and RUN it to see how they work. Also, we introduce the RND (random number generator) function. This function generates random numbers as explained below.

```
10 REM ***
20 REM *** HIGH LOW GAME ***
30 REM ***
40 CLS
50 PRINT "I AM THINKING OF A NUMBER
   BETWEEN Ø ANO 1ØØ"
60 PRINT "YOU MUST TRY TO GUESS MY NUMBER
   BY TYPING IN OIFFERENT NUMBERS"
70 PRINT "AND I WILL TELL YOU IF YOU ARE TOO
   HIGH OR TOO LOW OR IF YOU HAVE IT"
BØ RU = RNO (1ØØ)
9Ø PRINT : REM *** SKIP A LINE ***
1ØØ INPUT "ENTER GUESS"; G
11Ø IF G = RU THEN 1BØ : REM *** CORRECT ***
12Ø IF G > RU THEN GOSUB 16Ø
13Ø IF G < RU THEN GOSUB 17Ø
14Ø GOTO 9Ø
15Ø PRINT : REM *** SKIP A LINE ***
16Ø PRINT "IT WAS TOO HIGH" : RETURN
17Ø PRINT "IT WAS TOO LOW" : RETURN
18Ø FOR A = 234 TO 245 : SOUNO A,1 : NEXT A
    : REM *** SOUNO LOOP ***
19Ø PRINT "*** YOU GOT IT ***"
2ØØ PRINT : REM *** SKIP A LINE ***
21Ø PRINT "DO YOU WANT TO PLAY AGAIN"
22Ø INPUT I$
23Ø IF I$ = "YES" OR I$ = "Y" THEN BØ
24Ø IF I$ = "NO" OR I$= "N" THEN 26Ø
25Ø PRINT "WHAT" : GOTO 21Ø
26Ø PRINT
27Ø PRINT "OK, GOOOBYE"
28Ø ENO
```

In this game you are asked to pick a number from 1 to 1ØØ. The computer will tell you if your guess is too high or too low and, also, if your guess was correct. In line 8Ø the variable RU is the random number from 1 to 1ØØ. It uses a new function called RND. To get a range of random numbers, enter the highest random number you want generated in parentheses after RND. For example, if you wanted random numbers from 1 to 55, you would enter

```
RND(55)
```

The random numbers generated can be stored in variables, such as we did with RU. Line 11Ø checks to see if the number you entered was equal to the computer's random number RU; if not, the program will continue to line 12Ø where, if the number INPUTted is greater than (>) the random number RU, the program will do a different branch called a GOSUB. When the computer is at line 16Ø it will do the following: 1) it will print out it was to high, 2) it will see that there is a RETURN, and the computer will go back to where the GOSUB left off, at line 13Ø. (Now wasn't that SIMPLE?!) Even if we found that our number was greater than the random number, we still have to check to see if it is less than RU because the RETURN branched back to line 13Ø, right after the last GOSUB. You might be able to figure out what's happening in 13Ø. It's almost the same thing as in 12Ø, except if your number was less than RU the GOSUB would branch to 17Ø. Again, after printing out the message, the RETURN will go back to where the last GOSUB left off, line 14Ø. Line 18Ø is a SOUND loop

TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS

that will be explained in a later chapter. Here it serves as a congratulations for you if your guess is correct. Line 18∅ is where the computer branches if your guess is correct in line 11∅. In line 22∅ you're asked if you would like to play again, to which you reply either YES or NO, but did you notice that in 23∅ the computer checks to see if you INPUTted YES or Y. That's where the OR logical operator comes in. This logical operator will let you compare two or more acceptable answers you wish to have. It's the same in 24∅ except it checks for NO 'OR' N. If none of the answers were acceptable, line 25∅ tells you that your answer was bad and branches back to 21∅. Of course we always have to say "Goodbye" when people don't want to play any more. That's lines 26∅-28∅.

There still might be some questions about the GOSUB/RETURN statements. If so type the following and RUN it.

* * * IMPORTANT * * *

The following program must be spaced evenly. In the PRINT statements there are periods in place of spaces. This is for the convenience of not having to count the spaces in the line. Example line 3∅ has three periods (. . .). When typing in the line you should insert three spaces and then the message.

'O KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KID TO

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KI

* * * EXAMPLE * * *

(THE BOOK)
```
30 PRINT "... THE TRS-80 COLOR COMPUTER"
```
(YOU TYPE)
```
30 PRINT    " THE TRS-80 COLOR COMPUTER"
```

Now, remembering the periods are to be replaced by spaces, type in the following program.

```
10 CLS
20 GOSUB 200 : REM *** GO AND PRINT
    OUT 32 STARS ***
30 PRINT "... THE TRS-80 COLOR COMPUTER"
40 GOSUB 200
50 PRINT "..... THIS IS A DEMO OF THE "
60 GOSUB 200
70 PRINT "..... GOSUB/RETURN STATEMENT"
80 GOSUB 200
90 PRINT
100 PRINT
110 GOSUB 200
120 PRINT". THE COLOR COMPUTER IS THE BEST"
130 GOSUB 200
140 PRINT
150 PRINT "....... <><> THE END <><>"
160 GOTO 160 : REM *** INFINITE LOOP ***
170 REM ***
180 REM *** SUBROUTINE FOR PRINTING
     OUT 32 STARS ***
190 REM ***
```

```
2ØØ PRINT "********************************";
21Ø REM ***
22Ø REM *** BRANCH BACK TO WHERE THE LAST
    'GOSUB' LEFT OFF ***
23Ø REM ***
24Ø RETURN
```

Now let's discuss the flow of the program. In line 2Ø we GOSUB to line 2ØØ. Line 2ØØ PRINTs our 32 stars, then it goes down to 24Ø and branches back to where the last GOSUB left off in line 3Ø. The program will continue branching back and forth to the subroutine until it gets to line 16Ø and ends. Did you notice that instead of typing in several rows of stars, you had to do it just once by using the GOSUB/RETURN routine? The most important thing about this is we saved a great deal of typing and a bunch of valuable memory.

We finally made it. We're at the end of this chapter. Here are a few important things to remember. With the IF/THEN statements you can do complex testing of either numeric or string variables. Also GOSUB/RETURN statements are always good to use if a certain routine will be used more than once or twice.

TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KI

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KID TO

O KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

7

ARRAYS and READ/DATA

In this chapter we will discuss DIMs, ARRAYs, INPUTting ARRAYs and how to use READ/DATA statements. The best way to think about ARRAYs would be to think of them as a kind of variable. To understand the use of an ARRAY, type in the next program. (Remember to clear your memory with the NEW statement.)

```
NEW <ENTER>
10 CLS
20 PRINT "ENTER FIVE NAMES."
30 INPUT A$
40 INPUT B$
50 INPUT C$
60 INPUT D$
70 INPUT E$
80 CLS
90 REM ***
100 REM *** PRINT OUT THE NAMES ***
110 REM ***
120 PRINT "HIT 'ENTER' TO SEE LIST OF NAMES."
130 INPUT AA$
140 PRINT A$
```

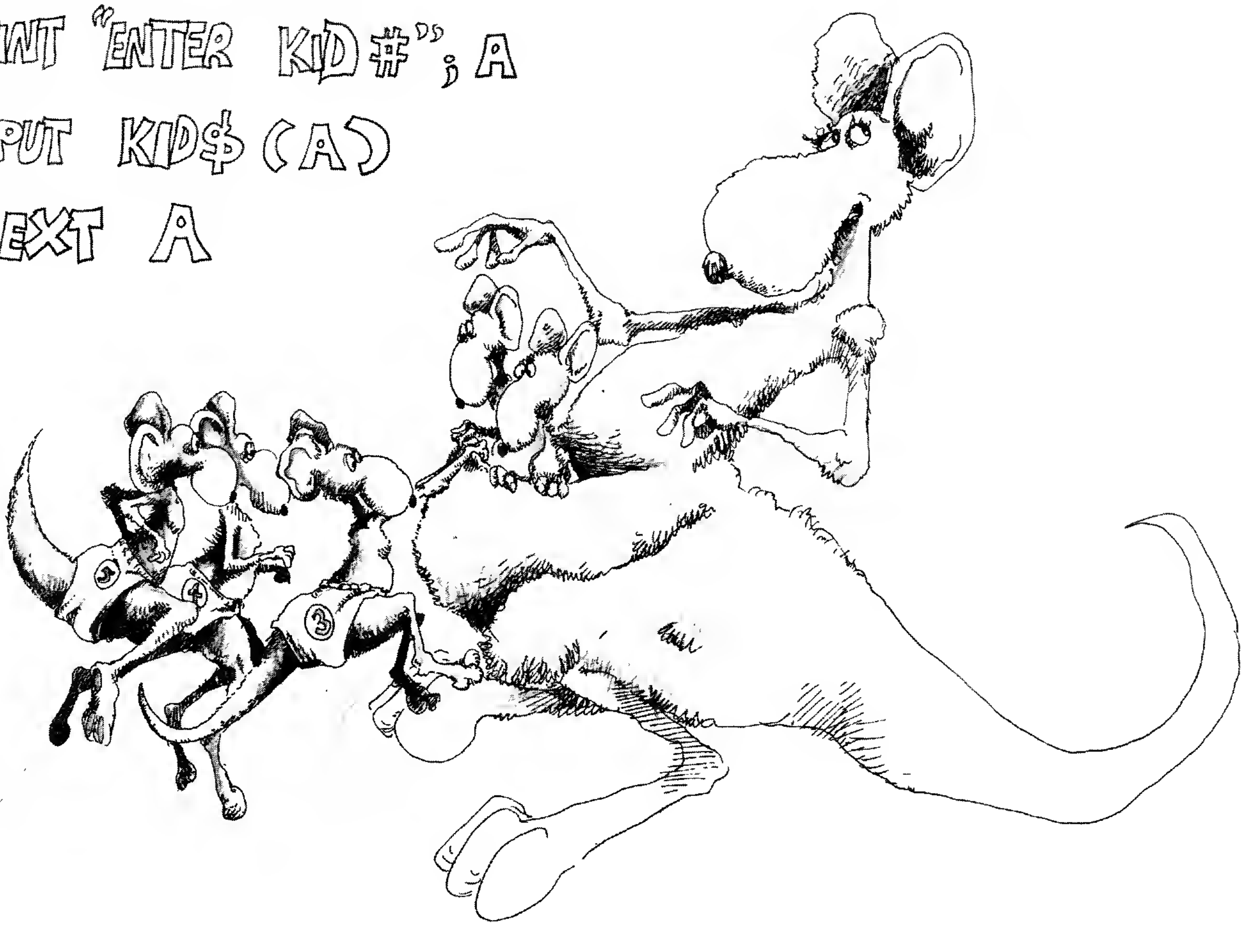
```
15Ø PRINT B$
16Ø PRINT C$
17Ø PRINT D$
18Ø PRINT E$
19Ø END
```

The program should be easy to understand, but it's not too good to use. If we have no more than one or two names or other types of data to be INPUTted, this method would be fine. How about trying to INPUT 1Ø or 15 names? Wouldn't that be a lot of INPUT statements? This is where ARRAYs can help us. Let's try the following program:

```
NEW <ENTER>
1Ø CLS
2Ø PRINT "ENTER FIVE NAMES"
3Ø PRINT : REM *** SKIP A LINE ***
4Ø REM ***
5Ø REM *** INPUT 5 NAMES ***
6Ø REM ***
7Ø FOR A = 1 TO 5 : REM *** SET LOOP ***
8Ø PRINT "ENTER NAME #" ; A
9Ø INPUT NAME$(A)
1ØØ NEXT A
11Ø PRINT "HIT 'ENTER' TO SEE NAMES"
12Ø INPUT ET$
13Ø PRINT : REM *** SKIP A LINE ***
14Ø REM *** PRINT OUT NAMES ***
15Ø FOR A = 1 TO 5
16Ø PRINT "NAME #" ; A ; "=" ; NAME$(A)
17Ø NEXT A
18Ø END
```

TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KID

```
10 FOR A= 1 TO 5
20 PRINT "ENTER KID#"; A
30 INPUT KID$(A)
40 NEXT A
```



TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KI

OK, now you might see the purpose of an array. If not, read carefully. In line 1Ø we set up the loop from 1 to 5. In 4Ø we ask for an INPUT for the string variable NAME$(A). At this point in the program A=1, so you're INPUTting NAME$(1). We will continue doing so until the loop is done (A=5). In line 9Ø we'll set up the loop from 1 to 5 again. Line 1ØØ will PRINT what number we're on and then PRINT out NAME$(1), and continue until it gets to A=5 or NAME$(5).

Let's say that you want to add more than just five names. You would adjust the loops to a higher number. EXAMPLE: FOR A = 1 TO 5 change to FOR A = 1 TO 2Ø. Go ahead and change line 2Ø to:

```
20 FOR A = 1 TO 20
```

RUN IT !!!

You should have run into a problem (BS ERROR). The computer defines this as a BAD SUBSCRIPT ERROR. When using strings or variables greater than 11, you must DIM (DIMension) that particular string or variable. The usual way to DIM an ARRAY is to DIM it to the highest number of times we want to use the ARRAY. For example, to DIM the array NAME$ to 15 we would enter

```
DIM NAME$(15)
```

This is because we will be using the string NAME$ 15 different times, and the computer must make room for the ARRAY data. Remember that DIM reserves a certain amount of memory when you specify how much you need by DIMension.

Let's fix the problem. Try this; insert the following line:

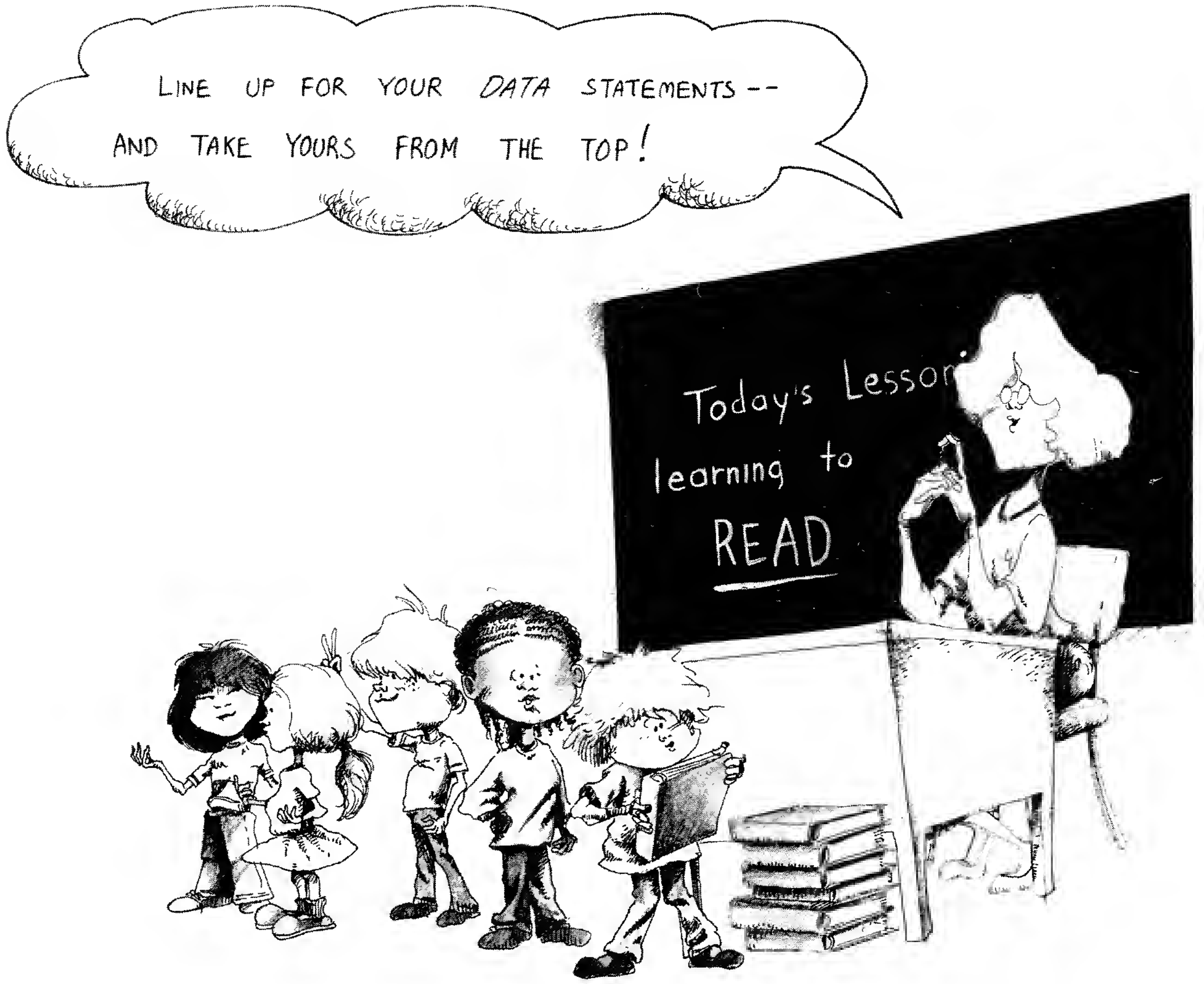
```
15 DIM NAME$(15)
```

You should have no trouble getting to the 15th INPUT.

What you have now is 15 variable names, NAME$(1) to NAME$(15). Look at the list below which compares regular string variables with array variables. We used an array of four in the example, but you can get the idea of how much easier it is to use arrays in certain applications instead of variables.

Regular	Array
A$	A$(1)
B$	A$(2)
C$	A$(3)
D$	A$(4)

With arrays we can generate the variable names using FOR/NEXT loops as we did in our example program. It saves a lot of time keying in variable names and makes our programs more flexible.


LINE UP FOR YOUR *DATA* STATEMENTS -- AND TAKE YOURS FROM THE TOP!
Today's Lesson
learning to
READ

It's time we learn two more BASIC statements. These next statements are the READ/DATA statements. Type in the following program and RUN it.

```
1Ø CLS
2Ø FOR A = 1 TO 5
3Ø READ D$ : REM *** GET DATA ***
4Ø PRINT D$
5Ø NEXT A
6Ø DATA MONSTERS, GOBLINS, WITCHES,
   GHOSTS, VAMPIRES
7Ø END
```

The program first sets the loop from 1 to 5 in line 2Ø. Then in 3Ø it READs the DATA from line 6Ø and stores it in D$. You must always separate each piece of DATA by a comma (,). This tells the computer where a new piece of data starts and ends. The first time through the loop the variable D$ reads MONSTERS, then GOBLINS and finally VAMPIRES. After the program is RUN, the values of D$ are as follows:

```
D$ = MONSTERS  WHEN A =1
D$ = GOBLIN    WHEN A =2
D$ = WITCHES   WHEN A =3
D$ = GHOSTS    WHEN A =4
D$ = VAMPIRES  WHEN A =5
```

In the next program we're going to tinker with tranferring data from DATA statements into string ARRAYs. This type of program would be useful when making a telephone-address file.

```
1Ø CLS
2Ø C=1 : REM ***SET ARRAY POINTER TO ONE ***
3Ø READ B$ : REM *** GET DATA ***
4Ø IF B$ = "END" THEN GOTO 8Ø
5Ø D$(C) = B$
6Ø C=C+1 : REM *** INCREASE COUNTER
   OF ARRAY ***
7Ø GOTO 3Ø
8Ø PRINT : PRINT "HIT 'ENTER' TO SEE NAMES"
9Ø INPUT ET$
1ØØ REM ***
11Ø REM *** PRINT OUT NAMES ***
12Ø REM ***
13Ø FOR A = 1 TO C-1
14Ø PRINT "DATA #" ; A ; "=" ; D$(A)
15Ø NEXT A
16Ø END
17Ø REM ***
18Ø REM *** PLACE YOUR INFORMATION BELOW ***
19Ø REM ***
2ØØ DATA DOG, CAT, COW, HORSE, PIG, GOAT, SHEEP,
    END
```

Notice we used the variable A in our FOR/NEXT loop in line 13Ø instead of C. It doesn't matter what variable names we used since all we want it to do is to generate the numbers 1 to 7. That's because our array variables are actually D$(1) to D$(7) and not D$(C) or D$(A). The C and the A just represent different numbers. Also, note how we used END as the last element in our DATA statement. When the computer READ "END", it stopped READing DATA into the array and jumped to the routine for PRINTing the array to the screen.

Since you've had so much experience with the DATA/READ statements, why not make that telephone-address file for yourself. (It's easy to do, but since your friends are probably something other than the barnyard characters we know, try putting names, addresses and telephone numbers in your DATA statements. Use separate arrays to READ the different parts of a name / address / telephone list.)

GOOD LUCK.

TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO K

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KID TO

O KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

8

MAKING SOUND

In the last chapter we saw how to use the DATA/READ statements and how useful they are. In this chapter we're going to experiment with DATA/READ and SOUND statements. There are two separate numbers that the SOUND statement requires. The first is PITCH, and it can range from 1 to 255. The second is the length of the note or sometimes called DURATION, which also ranges from 1 to 255. Try some examples. Just key in the examples and hit <ENTER>.

SOUND 1,255	The lowest and the longest note the computer can play.
SOUND 1,8	Same pitch, shorter duration.
SOUND 255,1	Highest pitch, shortest duration.

* * * NOTE * * *

The SOUND statement requires that both pitch and duration be from 1 to 255. If not you might get an ?FC ERROR. Go ahead and play around with sound for a few minutes and then type in the next program and RUN it.

PiTCH
CRACK!
RUMBLE

```
10 CLS
20 D=8 : REM *** DURATION TIME IS 8 ***
30 READ T : REM *** GET PITCH DATA ***
40 IF T = Ø THEN 8Ø
5Ø SOUND T,D
6Ø GOTO 3Ø
7Ø DATA 147, 159, 133, 5, 89, Ø
8Ø PRINT "....... CLDSE ENCOUNTERS .......... OF
   THE CDLDR CDMPUTER KIND"
9Ø RESTORE : REM *** RESET THE CDMPUTER TO
   THE START DF PITCH DATA ***
1ØØ D = D - 1 : REM *** DECREASE DURATIDN ***
11Ø IF D = Ø THEN 13Ø
12Ø GDTD 3Ø
13Ø PRINT : PRINT : REM *** SKIP TWD LINES ***
14Ø PRINT "          THEY'RE HERE !!!"
15Ø REM ***
16Ø REM *** TIMING LDDP ***
17Ø REM ***
18Ø FDR A = 1 TO 2ØØ : NEXT A
19Ø REM ***
2ØØ REM *** SDUND FDR THEY'RE HERE !!!"
21Ø REM ***
22Ø SDUND 159,4
23Ø SOUND 2ØØ,4
24Ø SOUND 185,8
25Ø END
```

In the program above we start out setting the SOUND duration to 8, then it READs the first piece of DATA from line 7Ø. Line 4Ø checks to see if the DATA was Ø. Look at the last piece of DATA in 7Ø, notice that it's a Ø. This will tell the computer that we're done playing the song and will allow us to continue with other parts of the program. If line 4Ø finds that yes, the data was a Ø then we will branch to 8Ø where it PRINTs out the message. Then in 1ØØ it decreases the variable D (duration) by 1. Line 11Ø checks to see if D is Ø, if not, the program will play the song again until D is equal to Ø. Every time the duration is decreased, the computer plays the song a little bit faster until D is Ø. When the duration finally becomes Ø we branch down to 13Ø and the program continues to 18Ø. Here, there is a timing loop which takes about one second. It then plays THEY'RE HERE!!!. With this type of program you can generate almost any type of desirable musical number.

The new statement we introduced in line 9Ø, RESTORE, resets the pointers to the beginning of the DATA statements. It works with any kind of READ/DATA statements, not just SOUND. It is useful if you want to READ in the same DATA more than once in a program.

The next program will help you generate a musical number. It will allow you to INSERT, EDIT, and PLAY MUSIC. Type in the following program and RUN it.

```
1Ø E = 3 : REM *** NUMBER OF NOTES ON THE
   SCREEN AT ONCE ***
2Ø DIM T(1ØØ), D(1ØØ) : REM *** RESERVE ROOM FOR
T & D ***
```

```
30 CLS
40 PRINT
50 PRINT " <1> INSERT OATA"
60 PRINT " <2> EDIT DATA"
70 PRINT " <3> PLAY MUSIC"
80 PRINT " <4> OISPLAY DATA"
90 PRINT" <5> EXIT"
100 GOSUB 140
110 ON S GOTO 190, 510, 710, 810 ,930
120 SOUNO 236, 8
130 GOTO 30
140 PRINT : PRINT ;
150 INPUT "ENTER SELECTION"; S
160 RETURN
170 CLS: PRINT@0," **** WHEN OONE TYPE 999 ****"
    : RETURN
180 FOR T = 1 TO 100 : IF T(T) = 0 THEN RETURN
    ELSE NEXT T
190 REM ***
200 REM *** INSERT OATA ROUTINE ***
210 REM ***
220 CLS
230 PRINT " <1> CONTINUE INSERTING OATA"
240 PRINT " <2> ALL NEW OATA"
250 GOSUB 140
260 ON S GOTO 280, 330
270 GOTO 190
280 CLS
290 PRINT "AT WHICH NUMBER OO YOU WISH TO
    CONTINUE"
300 INPUT C
```

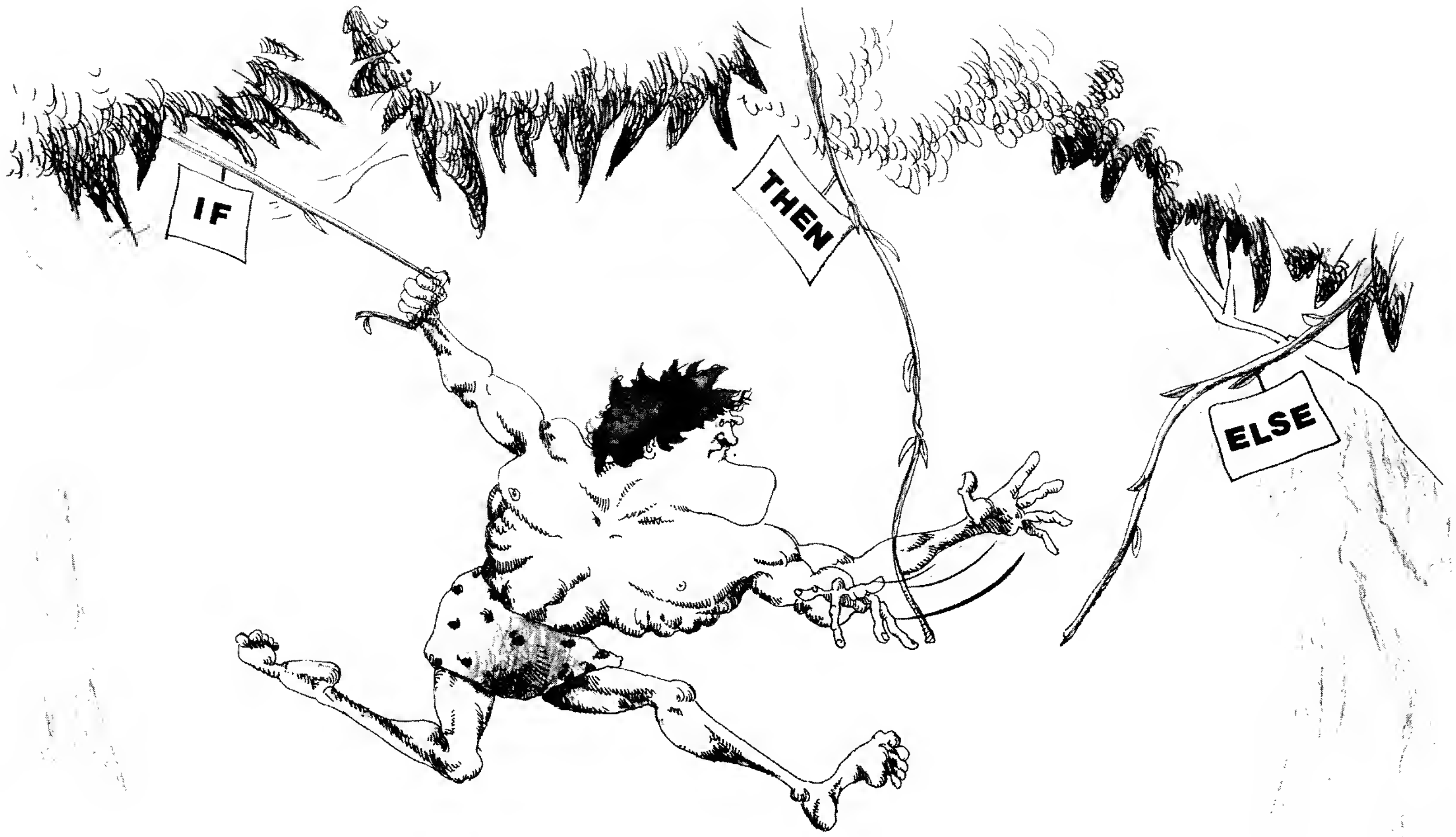
```
310Ø GOSUB 34Ø
320Ø GOTO 4Ø
330Ø C = 1
340Ø CLS
350Ø GOSUB 17Ø
360Ø GOTO 38Ø
370Ø PRINT "error redue" : SOUNO 245, 5
380Ø PRINT "data #" ; C
390Ø PRINT
400Ø INPUT "PITCH"; T
410Ø IF T=999 THEN 4Ø
420Ø INPUT "OURATION"; O
430Ø IF O = 999 THEN 4Ø
440Ø T(C) = T : O(C) = O
450Ø IF T(C) < 1 OR T(C) > 255 OR O(C) < 1
     OR O(C) > 255 THEN 37Ø
460Ø C = C + 1
470Ø IF E = C THEN GOSUB 17Ø ELSE 49Ø
480Ø E = C + 3
490Ø PRINT
500Ø GOTO 38Ø
510Ø REM ***
520Ø REM *** EOIT OATA ROUTINE ***
530Ø REM ***
540Ø CLS
550Ø PRINT "WHICH OATA LINE OO
     YOU WISH TO EOIT."
560Ø PRINT
570Ø INPUT "ENTER LINE NUMBER"; LN
580Ø IF LN > 1ØØ THEN 51Ø
590Ø PRINT "data #" ; LN
600Ø PRINT "PITCH=" ; T(LN)
```

```
61Ø PRINT "OURATION=" ; O(LN)
62Ø PRINT "*********************************" ;
63Ø PRINT "data #" ; LN
64Ø INPUT "PITCH" ; T(LN)
65Ø INPUT "OURATION"; O(LN)
66Ø PRINT
67Ø PRINT "OO YOU WANT TO EOIT ANY MORE"
68Ø PRINT
69Ø INPUT A$
7ØØ IF A$ = "YES" OR A$ = "Y" THEN 51Ø ELSE 4Ø
71Ø REM ***
72Ø REM *** PLAY OATA ROUTINE ***
73Ø REM ***
74Ø GOSUB 18Ø
75Ø T = T - 1
76Ø IF T = Ø THEN GOTO 4Ø
77Ø FOR P = 1 TO T
78Ø SOUNO T(P), O(P)
79Ø NEXT P
8ØØ GOTO 3Ø
81Ø REM ***
82Ø REM *** OISPLAY OATA ROUTINE ***
83Ø REM ***
84Ø CLS
85Ø GOSUB 18Ø
86Ø T = T - 1
87Ø FOR P = 1 TO T
88Ø PRINT"OATA #"; P ; "PITCH=" ; T(P) ; "OUR="; O(P)
89Ø NEXT P
9ØØ PRINT
91Ø INPUT " . . . . . HIT ENTER WHEN READY"; A$
92Ø GOTO 4Ø
93Ø ENO
```

With this program we're going to describe how the program *operates*, not how it *works*. When you first RUN the program it gives you a main menu. Let's say you wanted to generate music. You would enter #1. It then gives you another menu, and you should type in #2. Now, all you have to do is enter in the pitch/duration data. After each input you must hit <ENTER>. When you are finished, type in 999. This will send you back to the main menu. There you can either PLAY, DISPLAY NOTES, or EDIT DATA. When you're finished and have written down all of the PITCH and DURATION DATA, you can then put the DATA into a SOUND program.

There is a new statement we put into the program that you should understand. Notice lines 180, 470, and 700. There's something different about the IF/THEN statements. These are IF/THEN/ELSE statements. The IF/THEN/ELSE statement evaluates the IF/THEN condition, and if that condition is not met, it branches to the ELSE condition. Using ELSE in an IF/THEN sequence allows you to branch in two ways instead of one. The next little program shows you more clearly how it works.

```
10 CLS
20 PRINT "ENTER 1 OR 2"
30 INPUT N
40 IF N = 1 THEN GOTO 100 ELSE GOTO 200
50 END
100 PRINT "YOU PRINTED ONE"
110 END
200 PRINT "YOU PRINTED TWO OR MORE"
```


IF
THEN
ELSE

The next program allows you to put in the notes for your own program. Get some music, enter the note values in the DATA statements, and play your own sound.

```
10 CLS
20 REM *** PLACE YOUR TITLE .............. ***
   HERE ***
30 REM ***********************
40 READ T : REM *** GET PITCH ***
50 IF T = 8 THEN GOTO 100
60 READ D : REM *** GET DUR ***
70 SOUND T , D
80 GOTO 40
90 DATA :REM *** PUT YOUR OWN DATA HERE ***
100 REM "  * * * THE END * * *"
110 END
```

Well, that wraps up this chapter. Have fun making SOUNDs with your computer.

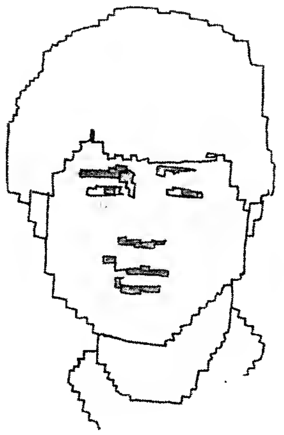
O KIDS TO KI

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KID TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KI

O KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

9 COCO GRAPHICS

This is one of our favorite chapters because it tells how to create pictures on the screen. We will show you how to make figures and give you some graphics so you will be able to create a game or some fascinating program! First, we will show you some of the statements used for graphics. Look at the program below to see what a program with graphics looks like. When you are finished looking at it, type it in.



```
1Ø FOR H=1 TO 45
2Ø SET(H,3,3)
3Ø SET(H,25,3)
4Ø NEXT H
5Ø GOTO 5Ø
```

Lines 1Ø and 4Ø set up a FOR/NEXT loop for H. In the SET statement, the first value is the horizontal position, the second the vertical position, and the third the color. In the program you see FOR H = 1 TO 45. This means that on the GRAPHICS SCREEN the computer will make a line from 1 all the way to 45 at vertical positions 3 (near the top) and 25 (near the bottom). What SET means is the computer "sets" a block on the screen where you tell it to. The loop first SETs a block at horizontal position 1, vertical position 3. Since the vertical position stays the same (3) and the value for H changes each time it goes through the loop, you get blocks lined up from horizontal position 1 to 45 all on vertical position 3. That's what the line really is — a line of blocks. The second line is made the same way except it is at vertical position 25. The color for the blocks is coded in the last number. In this case it is blue, or Code 3. A list of all the color codes is given later.

Line 5Ø says GOTO 5Ø so the program just sits there until you press BREAK. Leave line 5Ø out and see what happens. It should say OK at the top of the screen, which means that there is nothing more to do. The program is over and the computer asks, "OK, now what do you want me to do?" Since the graphics screen and locations are different from the text screen, we ought to take a look at it. You can have up to 64 horizontal (across) blocks and 32 vertical (down) blocks. You SET (horizontal, vertical, color) to place your block on the screen.

Graphic Screen

```
Ø..............32...............63

          <- Horizontal ->
     V
     e
     r
     t          H  V
16   i        X (32,16,C)
     c
     a
     l
31...............................63
```

In the above figure, the X is at horizontal location 32 and vertical location 16.

So far in this chapter you have learned how to use some of the graphics commands, such as SET and the FOR/NEXT loop to make a line. Before learning how to work with color, we will show you how to make vertical lines. Vertical lines are simple since they are made in the same way as horizontal lines, except you stack the graphic blocks on top of one another instead of side by side. Look at the program below to see how to make a vertical line. When you are finished looking at it, type it in.

```
1Ø FOR V=1Ø TO 26
2Ø SET(24,V,Ø)
3Ø SET(18,V,4)
4Ø NEXT I
5Ø GOTO 5Ø
```

On line 1Ø the program sets up a FOR/NEXT loop for V which means, you guessed it, Vertical. On line 2Ø it says SET(24,V,Ø). The 24 means how far horizontally it goes, and the V stands for vertical.

Right now it's about time you know how to use that third number in the SET statement. Well, it is the color of the blocks you are sending to your screen. In this case the blocks in line 2Ø are black. Look at the chart below to see all the colors your computer has. (What color vertical line will be made by line 2Ø?)

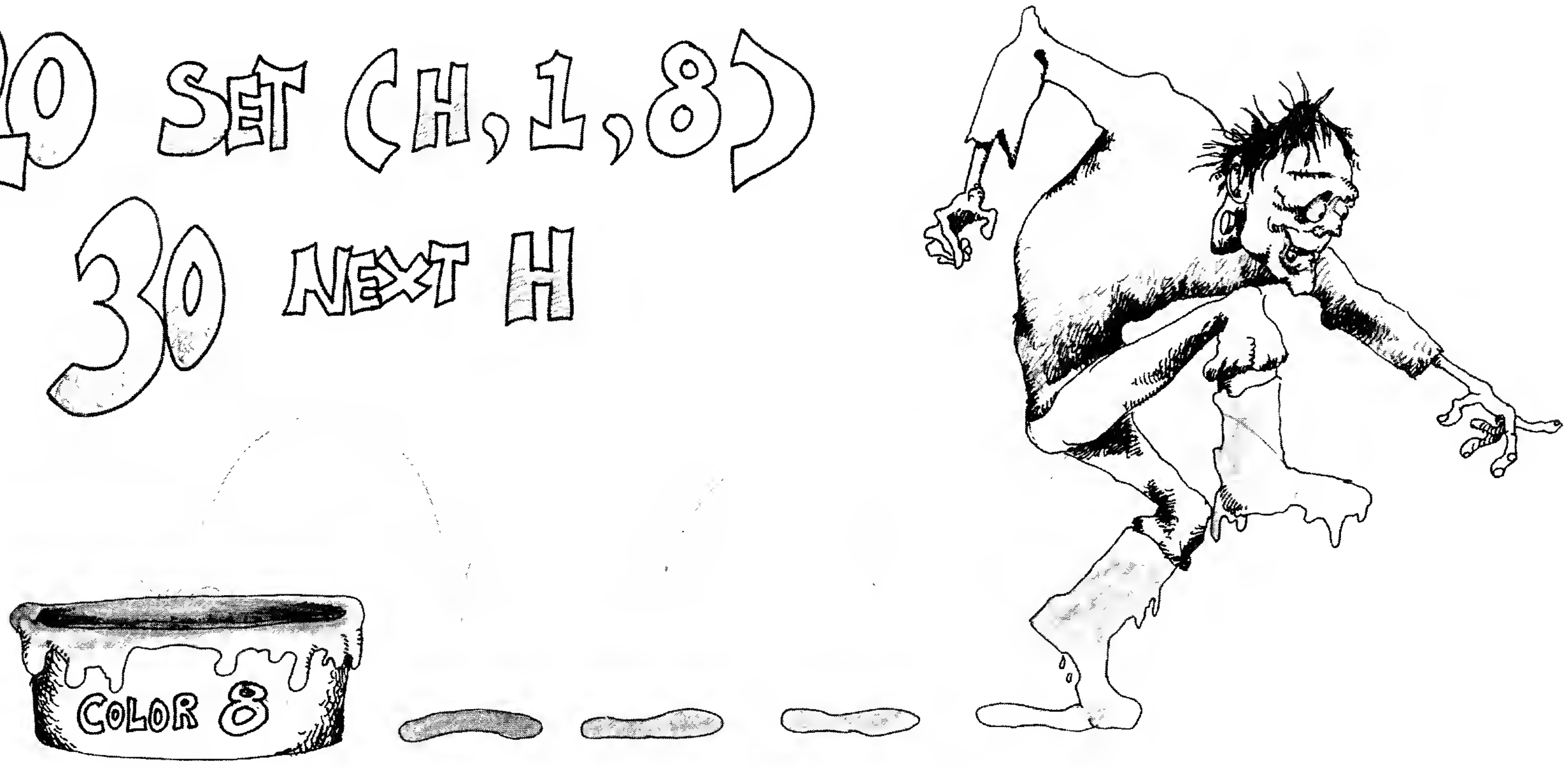
0=BLACK
1=GREEN
2=YELLOW
3=BLUE
4=RED
5=BUFF
6=CYAN
7=MAGENTA
8=ORANGE

Notice that there is a total of nine colors. Those colors can do a lot of different things besides making a line in different colors each time. Look at the program below to see how to use different colors in your programs. When you are finished looking at it, type it on your computer.

```
10 CLS(0) : GOSUB 100
20 CLS(1) : GOSUB 100
30 CLS(2) : GOSUB 100
40 CLS(3) : GOSUB 100
50 CLS(4) : GOSUB 100
60 CLS(5) : GOSUB 100
70 CLS(6) : GOSUB 100
80 CLS(7) : GOSUB 100
90 CLS(8) : GOSUB 100 : END
100 FOR HOLD = 1 TO 500 : NEXT HOLD
110 RETURN
```

'O KIDS TO KID TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KI

```
10 FOR H=1 TO 9
20 SET (H,1,8)
30 NEXT H
```

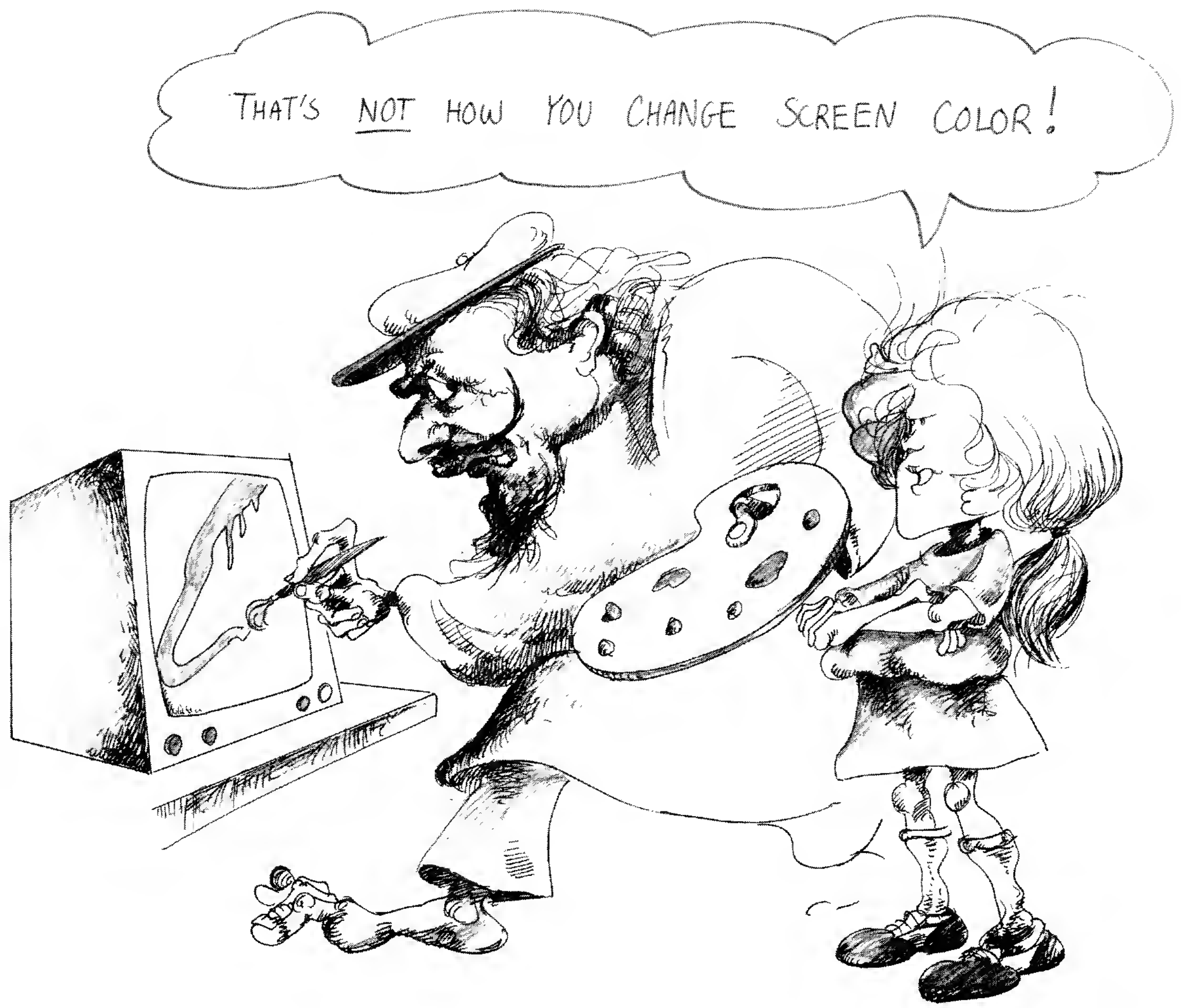


In the first chapter you learned that CLS means to clear the screen. When you place a number next to CLS in parentheses, it clears the screen to that color code. Run the program; it should have nine screens with a color from the program above. The FOR/NEXT loops, called HOLD, simply "hold" the screen color for a while so you can see it. If those loops were not in the program, it would clear the screens so fast you would be unable to see all the colors!

Now that you know how to make lines and color, it is time for, as the circus would call it, the grand finale. We will give you a program that will create a figure. Look at the program below to see what the program looks like and the similarities in making vertical and horizontal lines. When you are finished looking at it, type it in.

```
10 CLS(4)
20 FOR H=18 TO 24
30 SET(H,3,3)
40 SET(H,7,3)
50 NEXT H
60 FOR V=2 TO 7
70 SET(21,V,3) : NEXT V
80 FOR H=24 TO 25
90 SET(H,2,3)
100 SET(H,3,3)
110 NEXT H
120 FOR V=2 TO 3
130 SET(24,V,3)
140 SET(25,V,3)
150 NEXT V
160 GOTO 160
```

O KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KID TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO K



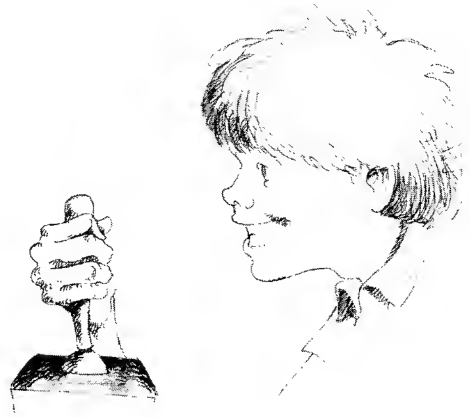
When you have finished typing it, type RUN. If you typed the program correctly, it should give you a blue figure on a red background. Now that you know how to do a lot of things with graphics, try making up some of your own characters.

In this book you are learning how to use graphics in many ways. When you grow up, maybe you can get a job making graphic programs. In the next chapter you will learn how to make a game.

10 HOW TO MAKE A GAME

In this chapter we'll discuss how to put together a simple game. All of the programs and discussions will be based on COLOR BASIC not EXTENDED COLOR BASIC. In developing a game, a major objective should be the purpose or goal of the game — such as catching bombs or shooting down the green invaders. Your game shouldn't be so hard that people won't win, or they just might lose interest. Of course, it certainly should not be too easy, enabling them to play for hours on end and not lose.

In the example programs, below, we will be using the JOYSTK function. The JOYSTK function works as follows: The computer says that from the TOP to the BOTTOM there are 64 different positions, from Ø through 63. This is also the same for the SIDE to SIDE position. If the joystick is in the middle, the computer will say the value is 32. The way we get the value is by using the simple statement X = JOYSTK(Ø), where X is the variable of the LEFT/RIGHT value of the joystick. Now we can get the UP/DOWN value part of the joystick by using Y = JOYSTK(1). To see this, type in the program below:



```
10 CLS
20 X = JOYSTK(Ø) : REM *** GET SIDE TO
   SIDE VALUE ***
```

```
30 Y = JOYSTK(1) : REM *** GET
   UP / DOWN VALUE ***
40 PRINT@ 96, "SIDE TO SIDE VALUE IS"; X
50 PRINT@ 128, "UP / DOWN VALUE IS"; Y
60 GOTO 20
```

If you want to get an AUDIO effect to demonstrate the joystick, add the following lines:

Note: when using the SOUND *statement, the two values have to be from 1 to 255. They could not be Ø. So, in line 4Ø we have to make sure that this requirement is fulfilled and make certain that the lowest value is 1, not Ø.*

```
40 IF Y = Ø THEN Y = 1 : REM *** Y CAN'T BE Ø ***
50 SOUND Y,1
60 GOTO 20
```

Let's really get fancy; add the following lines:

```
60 C = X
70 IF X > 8 THEN GOTO 20
80 CLS(C)
90 GOTO 20
```

Remember that the highest we can CLS is to 8, nothing higher.

Notice that the screen changes colors when you move the joystick slightly from the left side. These examples are to show you the basics behind the function. Use your imagination and come up with some other quick ideas. If you want to make the fire buttons work on your joysticks, type in the example. Use the right joystick.

```
1Ø CLS
2Ø 8 = PEEK (6528Ø)
3Ø PRINT B
4Ø GOTO 2Ø
5Ø END
```

There are two numbers that you should get, either 126 or 254, when pressing the RIGHT joystick button. Now use the LEFT joystick and see the difference in the numbers; it should be 125 or 253. This is the way we can tell which button was pressed first.

In the next program we'll show you how to draw with the joystick. When using this next program you can change the color of the graphic block by typing C and then typing in the desired color. The colors are from Ø to 8, using the same codes we saw in the last chapter. Here they are again so you can see what colors you're using.

< Ø > BLACK	< 5 > BUFF
< 1 > GREEN	< 6 > CYAN
< 2 > YELLOW	< 7 > MAGENTA
< 3 > BLUE	< 8 > ORANGE
< 4 > RED	

GOOD RESPONSE---

You can also clear the screen by pressing the joystick button. To erase a mistake, use color 1, which is green. However, be careful. You can draw right off the screen if you don't watch what you're doing.

```
10 CLS
20 CH = 128 : REM *** DRAWING COLOR.
   128=BLACK ***
30 X = JOYSTK(Ø)
40 Y = JOYSTK(1)
50 X = X/2 : REM *** CALCULATE WHERE TO PRINT
   ON THE SCREEN ***
60 Y = INT(Y/4) : REM *** SAME AS ABOVE ***
70 T = Y * 32 + X : REM *** SAME AS ABOVE ***
80 PRINT@T, CHR$(CH);
90 IF INKEY$= "C" THEN GOSUB 12Ø
   : REM *** CHECK TO SEE IF YOU WANT TO
   CHANGE THE COLORS ***
1ØØ IF PEEK(6528Ø) = 254 THEN CLS : REM *** SEE
    IF BUTTON IS BEING PRESSED ***
11Ø GOTO 3Ø
12Ø A$ = INKEY$
13Ø IF A$ = "" THEN 12Ø
14Ø V = VAL(A$)
15Ø CH = 128 + 16 * (V-1) + 15 : REM *** CALCULATE
    WANTED COLOR ***
16Ø IF CH>255 THEN CH=255 : REM *** ERROR
    TRAP AGAINST FC ERROR ***
17Ø RETURN
```

We're going to end this chapter with a simple game called CAR RACE. The game is simple and straightforward to use. When you first run the game, there is a timer, G. When G reaches 4ØØ the game breaks out of the 'MOVE CAR' routine and calculates your score. See if you can figure out exactly what every routine does, then improve or write your own version.

```
10 CLS
20 C = 271 : REM *** LOCATION OF YOUR
    CAR ON SCREEN ***
30 A$ = CHR$ (128)
40 PRINT @C, A$;
50 GOSUB 270 : REM *** GO PRINT OTHER CARS ***
60 PRINT @27, T; : REM *** PRINT HOW
    MANY TIMES HIT ***
70 SOUND 145,1
80 PRINT @ C-32, CHR$(143) ; : REM ERASE
    YOUR CAR ***
90 X = JOYSTK(0)
100 IF X < 20 OR X > 50 THEN GOSUB 150
     : REM *** DON'T MOVE IF THE JOYSTK IS
     IN THE MIDDLE ***
110 IF POINT (31+F,16) = 0 THEN GOSUB 320
     : REM *** CHECK TO SEE IF YOU HIT
     ANOTHER CAR ***
120 G = G + 1
130 IF G = 400 THEN 330 : REM *** IF G=400 THEN
     GO PRINT OUT THE SCORE ***
140 GOTO 40 : REM *** GO READ JOYSTK ***
150 IF X > 50 THEN 200 ELSE 210
```

```
170 REM ***
180 REM *** KEEP YOUR CAR WITHIN THE
     BOUNDARIES ***
190 REM ***
200 C = C + 1 : F = F + 2 : GOTO 22Ø
210 C = C - 1 : F = F - 2
220 IF F < -3Ø THEN F = -3Ø
230 IF F > 3Ø THEN F = 3Ø
240 IF C < 256 THEN C = 256
250 IF C > 286 THEN C = 286
260 RETURN
270 REM ***
280 REM *** ROUTINE FOR PRINTING OTHER CARS ***
290 REM ***
300 Z = RND(3Ø) : PRINT @48Ø+Z,
     CHR$(RND(4)+5*32-1)
310 RETURN
320 SOUND 245,1Ø : T = T + 1 : RETURN
330 FOR A = 1 TO 16 : SOUND RND(255),1 : NEXT A
340 CLS
350 PRINT "YOU WERE HIT" ; T ; "TIMES"
360 IF T = Ø THEN T = 2 : REM *** T=MUST ALWAYS
     BE GREATER THAN Ø, OR YOU'LL
     GET AN /Ø ERROR ***
370 PRINT "YOUR SCORE WAS" ; INT(G*2/T) ;
380 PRINT "OUT OF 4ØØ"
390 PRINT : PRINT
400 PRINT "DO YOU WANT TO PLAY AGAIN" ;
410 INPUT A$
420 IF A$ = "YES" OR A$ = "Y" THEN RUN
430 PRINT "CHICKEN!!!"
440 END
```

Well, we used a lot of different commands to make the game, and some we used are advanced. However, to make a good game, we sometimes have to get a little advanced. By now, you have learned most of the statements in the program; if you study the program carefully, you can get an idea of how it works. With practice, you can make your own games.

TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KI

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KID

O KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

11
HOW TO USE A PRINTER

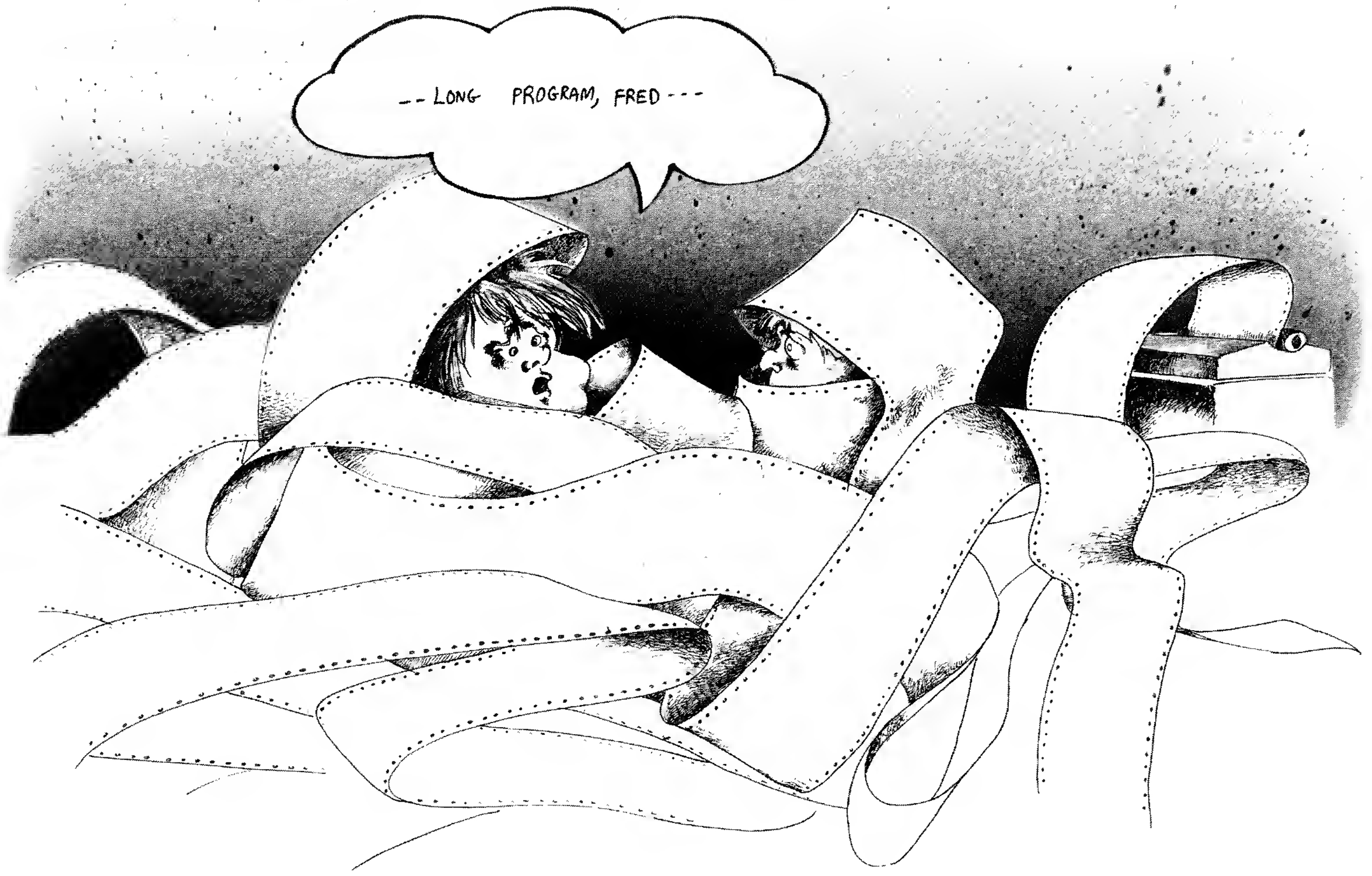
Printers are like computer typewriters, but they can do a lot more than a typewriter. There is a special set of printer commands to learn, which is what this whole chapter is all about. We are going to show you how to do some things, like how to make a LISTing of a program to your printer and how to print a sentence or two (or three or four or more!).

First we are going to show you how to LIST a program to your printer. The command for that is LLIST. Look at the program below to see how LLIST works. When you are finished looking at it, type it in.

```
10 PRINT "HI BILLY, HOW'S IT GOING?"
20 PRINT "FINE SAM, HOW'S IT GOING OVER THERE?"
30 PRINT "FINE, WELL I'LL SEE YOU TOMORROW"
40 PRINT"OK, BYE"
```

Now that you have finished the program, turn on your printer. Make sure it is ON LINE. If it is not ON LINE, flip the switch on your printer that will turn it ON LINE. (Some printers will have you choose SEL instead of ON LINE, but it's the same thing.) Now type

```
LLIST
```


-- LONG PROGRAM, FRED ---

The printer will print out the entire program including the line numbers and the statements. It will print everything that is in the program. All LLIST does is to LIST the program on the printer instead of on your TV screen. This may not seem too interesting, but when you are working on a long program trying to find bugs, it helps a lot to have a printer listing. It's called "hardcopy." Also, to send your friends a listing of your programs, the printed listing from your computer looks neat and you won't make mistakes in copying it.

Now that you know how to print out a LISTing (or LLISTing) to your printer we are going to show you how to print sentences, words, or letters or any other text you want to your printer. (We won't discuss printing out graphics since that depends on the type of printer you have and is pretty advanced.) Look at the next program to see how to print text to your printer.

```
10 D$ = "I OWN A DOG. HIS NAME IS MACHO."
20 PRINT #-2,D$
```

If it didn't work then look at your printer and make sure it is turned on and ON LINE. On your printer it should say,

```
I OWN A DOG. HIS NAME IS MACHO.
```

The whole secret to PRINTing to your printer instead of to your screen is in line 2∅. The statement PRINT #-2 instead of PRINT sends what would normally go to your screen to your printer. In this case D$ in line 1∅ is I OWN A DOG. HIS NAME IS MACHO. and that is what is sent

```
10 J$ = "MY BEST FRIEND'S NAME IS JONNY"
20 PRINT #-2,J$
```

to the printer. Now that we have that out of the way, feel free to change line 1∅ to whatever you want it to be. You can make a sentence or a paragraph or two.

Right now we will show you how to print lower case letters. Lower case letters are simple; look at the example below to see how to create lower case letters on your printer.

Make sure your printer is on and ON LINE. Now type RUN, and on your printer it should say,

```
My best friend's name is Jonny.
```

The only two letters that should be capitalized are the M in "My" and the J in "Jonny." To get lower case letters, press SHIFT <ZERO>. Instead of being black on green, the display is green on black. That's how you can tell if it is going to print upper or lower case on your printer. After you enter SHIFT <ZERO>, all the letters will be in lower case until you hold down the SHIFT key and press a key. The SHIFTED key makes it upper case. When you press SHIFT <ZERO> again, everything will be in upper case.

Our next program is a simple one to turn your printer into a typewriter. When you RUN the program it will clear the screen and put a question mark on the left side of the screen. Type in messages, and every time you hit ENTER, the text will be printed to your printer. Be sure not to put in too many letters — about 200 — before you press ENTER. You can use it to write letters to your friends. When you are finished, press "#" when the question mark appears and the program will end.

KID PROCESSOR

```
10 CLS
20 INPUT S$
30 IF S$ = "#" THEN END
40 PRINT
50 PRINT #-2,S$
60 PRINT #-2
70 GOTD 20
```

In this chapter you have learned how to make a printout of a program, print out sentences on the printer, and print out lower case letters on your printer. All you have learned in this chapter is how to print a lot of things to the printer. In the next chapter, you will learn how to do your homework on your computer.

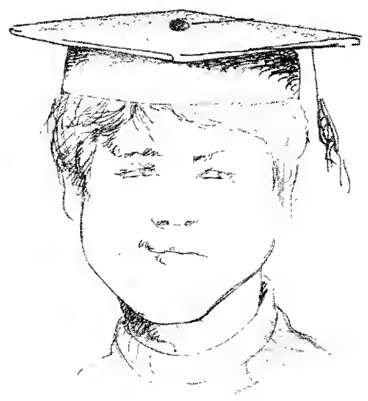
KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO K

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KID TO

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

12 DOING YOUR HOMEWORK ON THE COMPUTER

In this next chapter we'll show you how you can do your homework, so even your parents will want you to use your computer when doing homework. You say that's impossible!! Well, type the following program and you'll see why that's not impossible. (*Note: Where you see periods in the program listing, put spaces.*)



```
10 CLS
20 PRINT@ 102, “COCO MATH FUNCTIONS”
30 PRINT
40 PRINT “......<1>..ADDITION”
50 PRINT “......<2>..SUBTRACTION”
60 PRINT “......<3>..MULTIPLICATION”
70 PRINT “......<4>..DIVISION”
80 PRINT “......<5>..QUIT”
90 PRINT@ 422, “PICK A FUNCTION ”;
100 INPUT A
110 CLS
120 IF A = 1 THEN A$ = “ADDITION” : GOTO 210
130 IF A = 2 THEN A$ = “SUBTRACTION” : GOTO 210
140IF A=3 THEN A$=“MULTIPLICATION”:GOTO 210
150 IF A = 4 THEN A$ = “DIVISION” : GOTO 210
160 IF A = 5 THEN END
```

```
170 GOTO 10
180 REM ***
190 REM *** START OF MAIN ROUTINE ***
200 REM ***
210 PRINT@ 106, A$
220 PRINT
230 PRINT "ENTER THE FIRST NUMBER ";
240 INPUT B
250 PRINT "ENTER THE SECOND NUMBER ";
260 INPUT C
270 REM ***
280 REM *** FIND OUT WHICH MATH FUNCTION
    WANTED AND D=ANSWER ***
290 REM ***
300 IF A = 1 THEN D = B + C
310 IF A = 2 THEN D = B - C
320 IF A = 3 THEN D = B * C
330 IF A = 4 THEN D = B / C
340 PRINT
350 INPUT "WHAT IS YOUR ANSWER "; A
360 IF A < > D THEN 420 : REM *** CHECK IF
    YOUR WRONG ***
370 PRINT"YOU'RE RIGHT !"
380 PRINT
390 INPUT "DO YOU WANT TO TRY AGAIN "; A$
400 IF A$ = "Y" OR A$ = "YES" THEN 10
410 CLS : END
420 PRINT "SORRY, THE ANSWER IS "; D
430 GOTO 380
```

You should notice that the program is very short. We thought there would be no need to write four almost identical programs. We could have made the program out of three main GOSUB routines combined into one main program, but instead we did it a little differently.

Line 21Ø is the start of the main routine that asks for both numbers. You can also refer to the numbers as the NUMERATOR, which is the top or first number, and the DENOMINATOR, which is the bottom or last number. This really isn't too important here, but good to know. Mom and Dad should really appreciate a program such as this, especially since it wasn't available when they were in school. If you want to be a real wiz kid, why don't you try to make a spelling bee program? It will guarantee a 1ØØ% on your next test, provided that you study with the program.

WoRdPrOcEsSiNg

Have you ever wished you could be a fancy typist? And have great looking school reports? Well, wish no more. The power is right at your fingertips (with the help of your color computer). Most computers have the capabilities to use a word processor. For the color computer there are several different ones to choose from. Here are just a few: Color Scribe by Computerware, Telewriter-64 by Cognitec, and Text editor by Elite Software. All of these are excellent word processors. Color Scribe and Telewriter-64 were used in the making of this book. Word processors (W/P) are generally very easy to use and understand. This is called being User Friendly. The W/P will allow you to do such things as realign text. What this means is it will either fill in or take out text and insert it elsewhere to properly align the text.

* * * E x a m p l e * * *

This is a dictionary definition of the word Computer: a programmable electronic device that can store, retrieve, and process data.

Definition of the word Memory: for a computer it can be inserted and stored and from which it may be extracted when wanted; for example, 16K bytes is equal to 16384 (16 X 1Ø24) pieces of information.

* * * After using the align function * * *

This is a dictionary definition of the word Computer: a programmable electronic device that can store, retrieve, and process data.

Definition of the word Memory: for a computer it can be inserted and stored and from which it may be extracted when wanted; for example, 16K bytes is equal to 16384 (16 X 1Ø24) pieces of information.

There certainly is quite a difference from the non-aligned definitions to the aligned defintions. Of course this is a little exaggerated but shows you a point about the align function. We'll now give you a short list of functions that most W/P have for writing letters, documentation, and other written materials:

<1> Word Wrap; when at the end of a line and you are still typing a word, the Word Wrap function will bring the entire word down to the next line.

<2> Block Delete; will delete a block of text that you do not want.

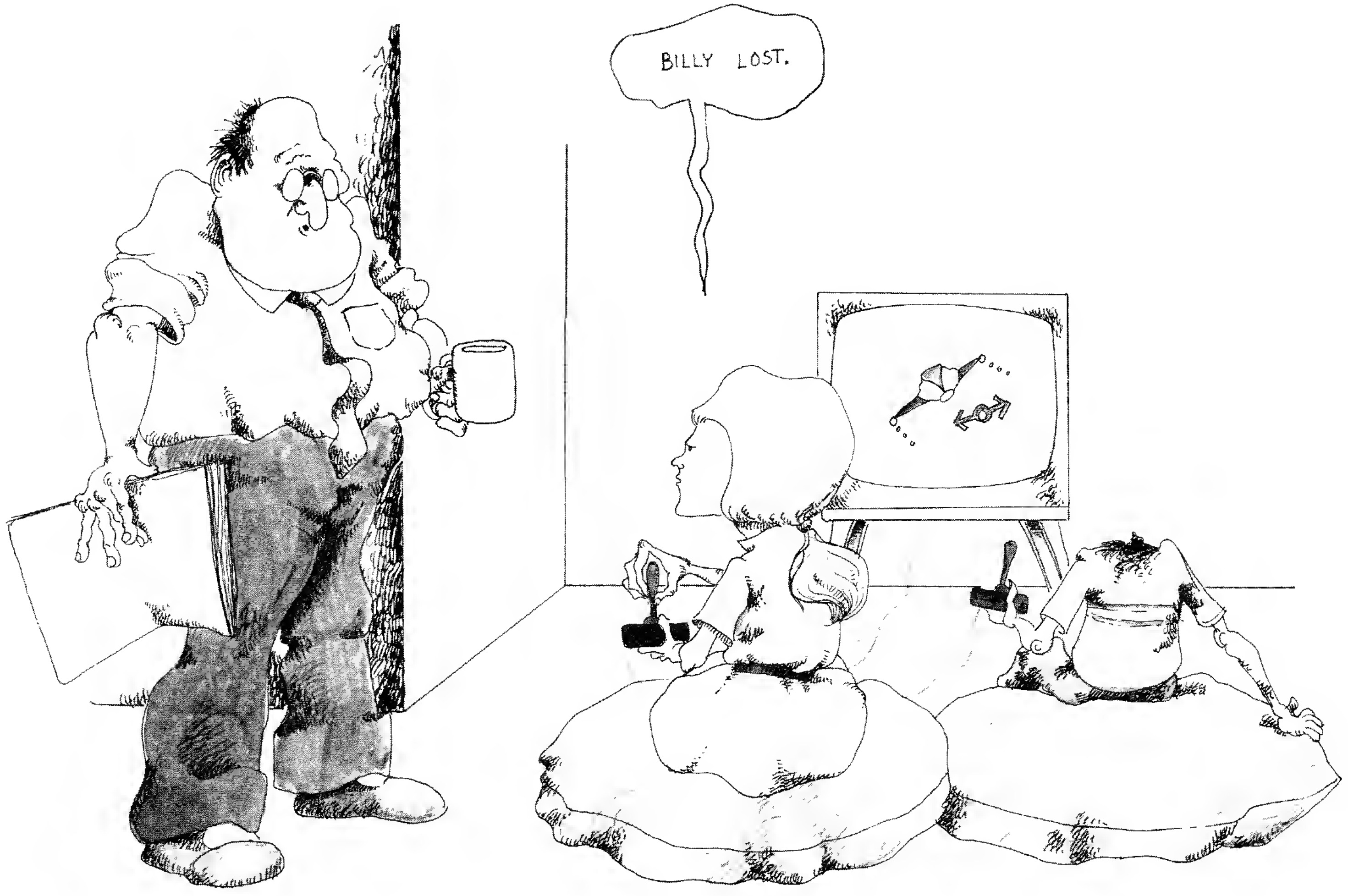
<3> Copy Block; will copy a block of text and place it where indicated by the user.

<4> Line Delete; delete a single unwanted line.
<5> Find; the W/P will find the character you indicate.
<6> Global; searches for a character and replaces it.
<7> Save; save text to tape or to disk.
<8> Load; load text from tape or disk.
<9> Append; tacks on a file of text to the end of the existing file in memory. (combining letters)
<1Ø> Print; prints out the file in memory to a printer.

WRITTEN ASSIGNMENTS ON A WORD PROCESSOR

Whenever we're given a written assignment like a report or grammar homework, after we write the assignment, our parents go over it and check spelling and English. Then we have to write the whole thing over again. With a word processor, though, it's really simple. All we have to do is to go back and change the mistakes, and then send it to the printer. The printer will do it as many times as you want until it's correct. You don't have to re-type anything except the mistakes. It helps you concentrate on correct grammar and spelling instead of all the work in re-doing the whole paper. Also, when the paper is turned in, it looks much better and clearer than a hand-written assignment.

There are many, many more such commands as these that will enable you to produce great looking material. Through practice you will find that the word processor can be a very useful tool in doing just about anything.


BILLY LOST.

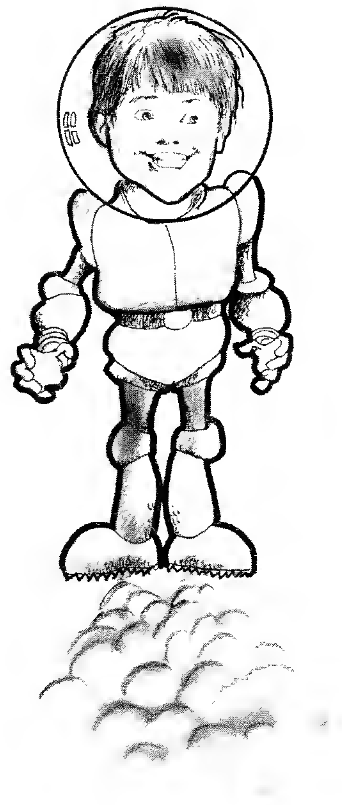
13

OUR FAVORITE PROGRAMS/GAMES

In this chapter we'll discuss different companies' products for the Color Computer and other material related to it. We'll also describe a few games and then give a listing of the most popular games to date.

Have you ever wanted to explore a strange house and encounter black magic? If so, BLACK SANCTUM is for you. This adventure game is quite fun and tough to beat, but with time you should conquer the evil that dwells there. Another excellent game is PYRAMID. Both of these games are from Mark Data Products and are each under $2Ø.ØØ. There are many more adventure games also available from different companies. If you want more information, look in a few Color Computer magazines.

Next we'll describe some fast action arcade type games. If you think it would be nice to be a space warrior, then ZAXXON is, of course, for you. ZAXXON is a game developed by Steve Bjork from Datasoft Inc. You first start out flying around empty space for a bit, then you come up to an island that has walls all around it. All of a sudden you see a base tank fire laser shots at you, so you respond by blowing it up first (you hope). After lots of shooting you come face to face with the robot Zaxxon.



This guy is a toughy, but he can be destroyed. After a while the game repeats and gets a little tougher each time. This game has just about everything the arcade ZAXXON has and more.

Another game that is very good is THE KING by Tom Mix Software. This game again is almost exactly like the real arcade game. There are four different kinds of mazes — the beams, the pegs,the elevator, and the pie factory. The graphics are very colorful and the program very well done.

DEFENDER by Fishertronics is another awesome game with excellent graphics much like the real arcade game by Williams Electronics. TRAPFALL by Tom Mix Software is much like PITFALL from the ATARI ACTIVISION cartridge but it is better. If you like PITFALL, you will love TRAPFALL.

BERSERK by Tom Mix Software is very similar to the real arcade game as well. You travel through mazes trying to destroy every robot in sight. Watch out, beware of Evilotto. He is indestructible. Another creative game is GHOST GOBBLER from Spectral Associates. This is a very good PAC-MAN style game. It has very good graphics much like the arcade game. Eat your way through mazes, but watch out for the ghost monsters. LANCER from Spectral Associates is a great game, too. This is like the arcade game, JOUST. You try to hit flying men on great flying birds. Watch it, if you don't keep on flapping you may fall into the hot molten lava below.

COLORPEDE by Intracolor is almost exactly like the arcade game CENTIPEDE. You are trapped in a kind of maze of mushrooms, spiders, fleas, scorpions, and the dreaded COLORPEDE. The spiders make you nervous and take away mushrooms. If you shoot too many mushrooms, a flea will come down and make more of them. The scorpions will hit mushrooms and make them poisonous and if the COLORPEDE hits the mushroom, the COLORPEDE will come straight down and turn just before it hits the bottom of the screen.

A game that has excellent graphics and is very well written is ASTROBLAST by Mark Data Products. This game is like the arcade game. You travel through space destroying enemy ships and try to get reloaded with fuel by shooting the mother ship. This game is very good. The last, but not least, game is THE FROG by Tom Mix Software. This game is like the arcade game FROGGER by Sega. Hop across the road avoiding cars, hop onto logs and turtles, and beware of the crocodile. These are just a handful of Color Computer games that are available.

*** Aardvark Software ***
Wizards tower
Golf
Haunted House
Quest
Venturer
Dungeons of Death

*** Computerware ***
Doodle Bug
Moon Hopper
Grand Prix
Bloc Head
Sharks Treasure

Pac Attack
Rail Runner
Megapede
Nerble Force

*** Cornsoft ***
Scarfman

*** Datasoft ***
Zaxxon
Moon Shuttle

*** Elite ***
Zaksund

*** Mark Data ***
Haywire
Calixto Island
Space Raiders
Glaxxons
Cave Hunter
Astro Blast
Black Sanctum

*** Med Systems ***
Phantom Slayer

*** Spectral Associates ***
Space Invaders
Space Racs
Planet Invasion
Galax Attax
Ghost Gobbler

*** Tom Mix ***
Trapfall
Yaazee
Katerpillar Attack
The Frog
Space Shuttle
Protectors
The King

To make your Color Computer useful for programming, companies have made several different tools (utilities) for making BASIC programming much easier and faster. Utilities include Business, Data Communications (using the phone line to communicate with another computer), Disk Operating Systems and Education. Here is a small list of neat utilities.

* * * BUSINESS * * *

*** Branrex,Inc. COLOR SOFTware Services div. ***
General Ledger
Small Business Accounting
Management Skill
Accounts Receivable
Depreciation
Loan Analysis
Annunity
Expense Account Diary
Stock Analyzer

*** Cognitec ***
Telewriter-64 (Word Processor)

*** Computer Systems Center ***
Dynacalc

*** Computerware ***
Color Scribe (Word Processor)

*** MPP Graphics ***
Stock Portfolio Management
Check Book

*** Softlaw Corp. ***
VIP Calc
VIP Database
VIP Speller
VIP Writer

*** Spectral Associates ***
Business Analysis

*** Radio Shack ***
Personal Finance
Spectacular
Investment Analysis

* * * DATA COMMUNICATIONS * * *

*** Computerware ***
The Color Connection

*** Micro Works ***
Microtext

*** Softlaw Corp. ***
VIP Terminal

*** Eigen Systems ***
ColorCom/E

*** Martin Consulting ***
Colorterm 1.1

*** DSL Computer Products ***
Color DFT

*** Double Density Software ***
Color Term +plus+

* * DISK OPERATING SYSTEMS & other utilities * *

*** Arizin ***
Colorkit (Excellent tool. Has many bells,
whistles and aids to help program in Basic.)

*** Eigen Systems ***
Disk Basic Aid

*** Frank Hogg Laboratory ***
FLEX (Powerful Disk Operating System)

*** Micro Works ***
Macro-8ØC (disk based assembler and monitor)
SDS8ØC (assembler and monitor)
Cbug (monitor)
Source Generator (disassembler)

*** Softlaw Corp. ***
VIP Disk-Zap

*** Prickly-Pear Software ***
The Disk Manager (recover a crashed disk and more)
The Disk Master (Speed check, disk map, purge files and more)

*** Tom Mix Software ***
Disk to Tape
Tape to Disk
The Fixer (Converts non-Disk software $Ø6ØØ, to Disk Software)

* * * EDUCATION * * *

*** Computer Island ***
Circus Adventure
School Maze Adventure
Reading Aids 4-Pak
Foreign Language Baseball (Spanish, French, Italian)
Money-Pak
Beyond Words

*** Radio Shack ***
Color Computer Learning Lab
Vocabulary Tutor
Color Logo
Color Pilot

Note: Some Radio Shack Computer Centers offer a BASIC computer class for the Color Computer. Check at your local computer center for more details.

*** Spectral Associates ***
Mathdrill
Spelling Master
Typing Teacher
Geography Pac
Foreign Language games
Dollars and Sense
Circus (help developing reading skills)

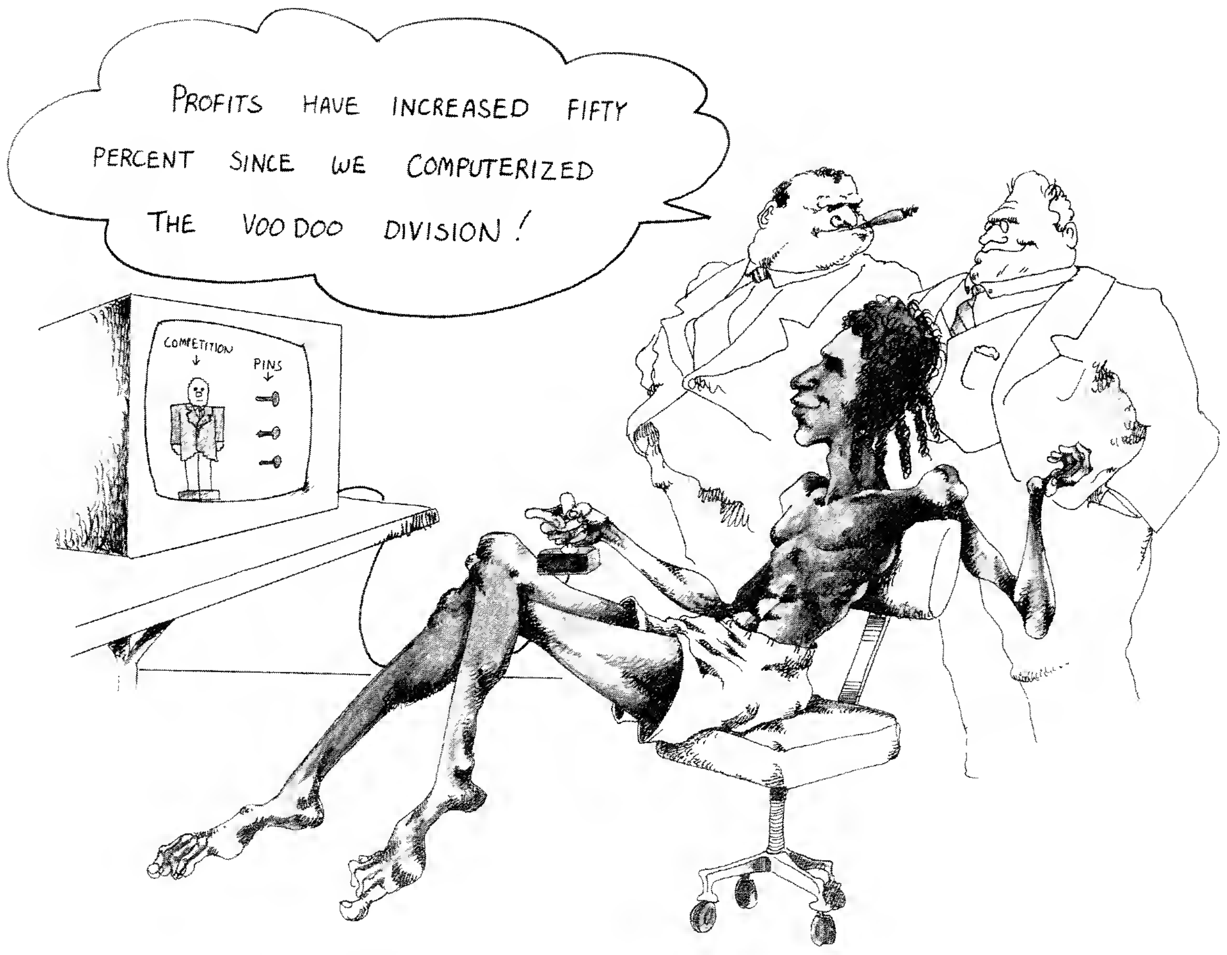
*** Sugar Software ***
Galactic Hangman

All of these products are made specifically for the Color Computer. The best way to find out more is to write to the companies. The easiest way to get their addresses is to purchase any number of Color Computer magazines out. Here is a list of such magazines:

Rainbow
Color Computer News
Color Computer
Hot CoCo

There are also many other TRS-8Ø Computer magazines out, and most include programs and material for the Color Computer. So there is a great deal of support for the CoCo Home Computer if you know where to look.

O KIDS TO KID


PROFITS HAVE INCREASED FIFTY PERCENT SINCE WE COMPUTERIZED THE VOODOO DIVISION!
COMPETITION
PINS

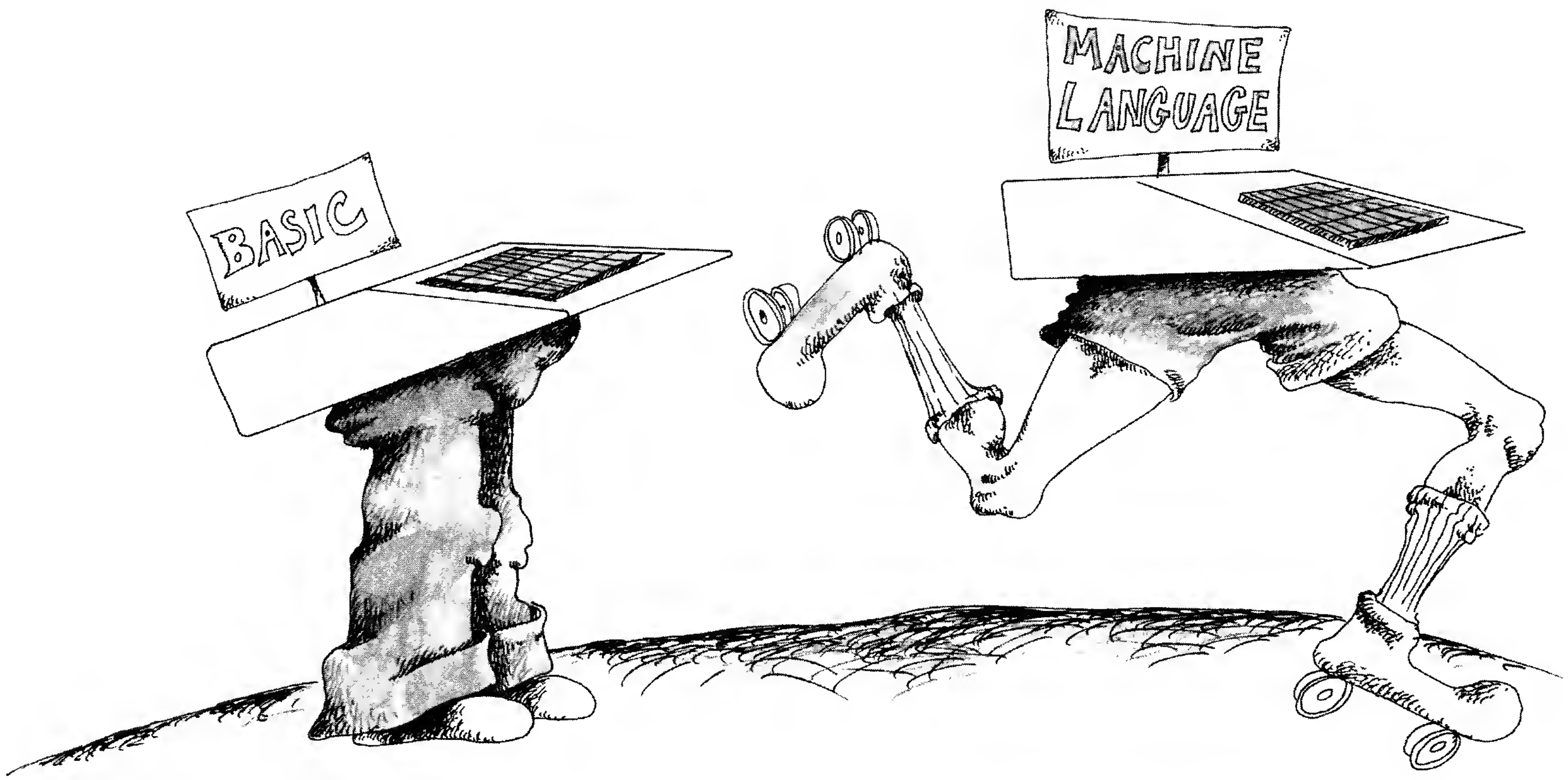
TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KI

TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KI

14 COCO PROGRAMS

We've filled this chapter with several different types of programs for you to learn from and use on your Color Computer. There will be some advanced programs, but don't worry about understanding them right away. Machine language and advanced programming techniques are used to add speed to some of the programs. The programs that have machine language subroutines are: 1) Error Sound and 2) Key Stroke. After mastering BASIC you may want to move on and look into machine language programming or other types of languages.

The following two programs should give you some ideas about how to put together interesting and colorful graphics using the low resolution function of the Color Computer. The first program shows you the concept on using DATA statements for storing graphic images. Where there is a 1 the computer will plot (SET) a dot on the screen. When there is a Ø the computer skips to the next piece of DATA and doesn't plot on the screen. You should use the first program to help yourself fully understand the process of transferring the image to the screen. Good Resolutions !!!


BASIC
MACHINE LANGUAGE

```
10 CLSØ
20 X = Ø : Y = 8
30 FOR H = X TO Y
40 READ PLOT
50 IF PLDT = Ø THEN 7Ø
60 SET(H,V,5)
70 NEXT H
80 V = V + 1
90 IF V < 7 THEN 3Ø
100 GOTD 1ØØ
500 DATA Ø,Ø,1,1,1,1,1,Ø,Ø
510 DATA 1,1,1,1,1,1,1,1,1
520 DATA Ø,Ø,1,Ø,1,Ø,1,Ø,Ø
530 DATA Ø,1,1,1,1,1,1,1,Ø
540 DATA Ø,Ø,1,Ø,1,Ø,1,Ø,Ø
550 DATA Ø,1,Ø,Ø,Ø,Ø,Ø,1,Ø
560 DATA 1,Ø,Ø,Ø,Ø,Ø,Ø,Ø,1
```

* * * * * *

```
10 CLSØ
20 PRINT@ 459, "INVASIDN";
30 X = Ø : Y = 8 : REM *** FIRST CODROINATED
    FDR INVADERS ***
40 REM ***
50 REM *** PRINT THE INVADER ***
60 REM ***
70 FOR H = X TD Y
80 READ PLOT
90 IF PLDT = Ø THEN 2ØØ
100 SET( H + 9, V, C )
110 SET( H + 9, V + 9, C )
```

```
120 SET( H, V + 9, C )
130 SET( H, V, C )
140 SET( H + 18, V, C )
150 SET( H + 18, V + 18, C )
160 SET( H, V + 18 , C )
170 SET( H, V, C )
180 SET(H + 9, V + 18, C )
190 SET ( H + 18, V + 9, C )
200 NEXT H
210 V = V + 1
220 IF V < 7 THEN 7Ø
230 H = Ø : V = Ø : RESTDRE
240 C = C + 1 : REM *** INCREASE COLOR ***
250 IF C = 9 THEN C = Ø
260 X = 31 : Y = 39 : REM *** SECOND COORDINATED
     FOR INVADERS ***
270 GOTO 7Ø
280 REM ***
290 REM *** DATA FOR INVADER ***
300 REM ***
500 DATA Ø,Ø,1,1,1,1,1,Ø,Ø
510 DATA 1,1,1,1,1,1,1,1,1
520 DATA Ø,Ø,1,Ø,1,Ø,1,Ø,Ø
530 DATA Ø,1,1,1,1,1,1,1,Ø
540 DATA Ø,Ø,1,Ø,1,Ø,1,Ø,Ø
550 DATA Ø,1,Ø,Ø,Ø,Ø,Ø,1,Ø
560 DATA 1,Ø,Ø,Ø,Ø,Ø,Ø,Ø,1
```

This program is another one of those shoot 'em up games that consists of a base (you) and the flying saucers. The game is somewhat crude but it shows that it is possible to have a fun action game in Color Basic. You may change the characters of the base and of the ships by modifying the strings to different character values. (Look in the back of your Color Basic manual, pg. 276, to calculate your own character code.) A$(1) - A$(4) are the string arrays for the ships, and BASE$ is for the base. To play the game, use the right and left arrow keys to move and the space bar to shoot.

```
10Ø CLS
20Ø BASE$ 3 CHR$(143) + CHR$(199) +
    CHR$(2Ø7) + CHR$(2Ø3) + CHR$(143)
3Ø A$(1) = CHR$(143) + CHR$(183) +
    CHR$(187) + CHR$(143)
4Ø A$(2) = CHR$(143) + CHR$(!9Ø) +
    CHR$(189) + CHR$(143)
5Ø A$(3) = CHR$(143) + CHR$(183) + CHR$(191) +
    CHR$(187) + CHR$(143)
6Ø A$(4) = CHR$(143) + CHR$(195) + CHR$(195) +
    CHR$(195) + CHR$(143)
7Ø LEVEL = RND(8) : T = T + 1
8Ø IF T = 25 THEN 47Ø
9Ø SHIP = RND(4)
1ØØ ON RND(2) GOTO 11Ø, 18Ø
11Ø FOR L = Ø TO 26 STEP RNO(3)
12Ø PRINT@ LEVEL * 32 - 32 + L ,"  "; A$(SHIP); " ";
13Ø GOSUB 23Ø
14Ø PRINT@ 448 + AOO, BASE$;
15Ø NEXT L
16Ø PRINT@ LEVEL * 32 - 31, "          ";
17Ø GOTO 7Ø
18Ø FOR L=26 TO Ø STEP -(RNO(3))
19Ø GOSUB 23Ø
```

```
200 NEXT L
210 PRINT@ LEVEL * 32 - 31, "          ";
220 GOTO 70
230 IF PEEK(344) = 247 THEN AOD = ADD + 1
    : IF AOO > 28 THEN ADO = 28
240 IF PEEK(343) = 247 THEN ADD = AOO - 1
    : IF AOD < 1 THEN ADD = 1 .
250 IF PEEK(345) = 247 THEN 290
260 PRINT@ LEVEL * 32 - 32 + L, " "; A$(SHIP); " ";
270 PRINT@ 448 + AOD, BASE$;
280 RETURN
290 FOR FR = 13 TO LEVEL - 1 STEP - 1
300 WH = FR * 32 + ADO + 2
310 PRINT@ WH, CHR$(128)
320 PRINT@ WH, CHR$(143)
330 NEXT FR
340 CO = CO + 1
350 LV = LEVEL * 32 - 32 + L + 4
360 FOR XX = 1 TO 3
370 IF WH - 3 + XX = LV THEN 430
380 NEXT XX
390 FOR XX = 1 TO 3
400 IF WH - 2 + XX = LV THEN 430
410 NEXT XX
420 GOTO 260
430 SOUND 236, 1
440 PRINT@ LEVEL * 32 - 31, "          ";
450 NH = NH + 1
460 GOTO 70
470 FOR A = 1 TO 5
480 SOUND 45, 1 : SOUNO 95, 1
490 NEXT A
```

```
500 CLS
510 PRINT "SHOTS FIRED="; CO
520 PRINT "NUMBER HIT="; NH
530 PRINT "SHIPS MISSED="; T - NH
540 PRINT
550 PRINT "DO YOU WANT TO PLAY AGAIN";
560 INPUT B$
570 IF B$ = "Y" OR B$ = "YES" THEN RUN
580 PRINT " CHICKEN" : END
```

This next program will turn your screen into inverse colors. Let's say you have an opening title to a program — this short program would turn all that's on the screen to inverse characters. Then, if you wanted, you could change the screen back to normal display.

```
10 CLS
20 PRINT "THIS PROGRAM WILL EITHER TURN THE
   SCREEN INVERSE OR BACK TO"
30 PRINT "NORMAL. THIS WOULD BE GOOD FOR "
40 PRINT "THE OPENING TITLES OF YOU "
50 PRINT "OWN PROGRAMS"
60 PRINT@ 360, "INVERSE/NORMAL"
70 PIC = 0 : REM *** INVERSE ***
80 GOSUB 10000
90 PIC = 1 : REM *** NORMAL ***
100 GOSUB 10000
110 END
10000 FOR D = 1024 TO 1535 : REM *** BEGING
    SCREEN ADDRESS AND ENDING SCREEN
    ADDRESS ***
10010 A = PEEK(D)
```

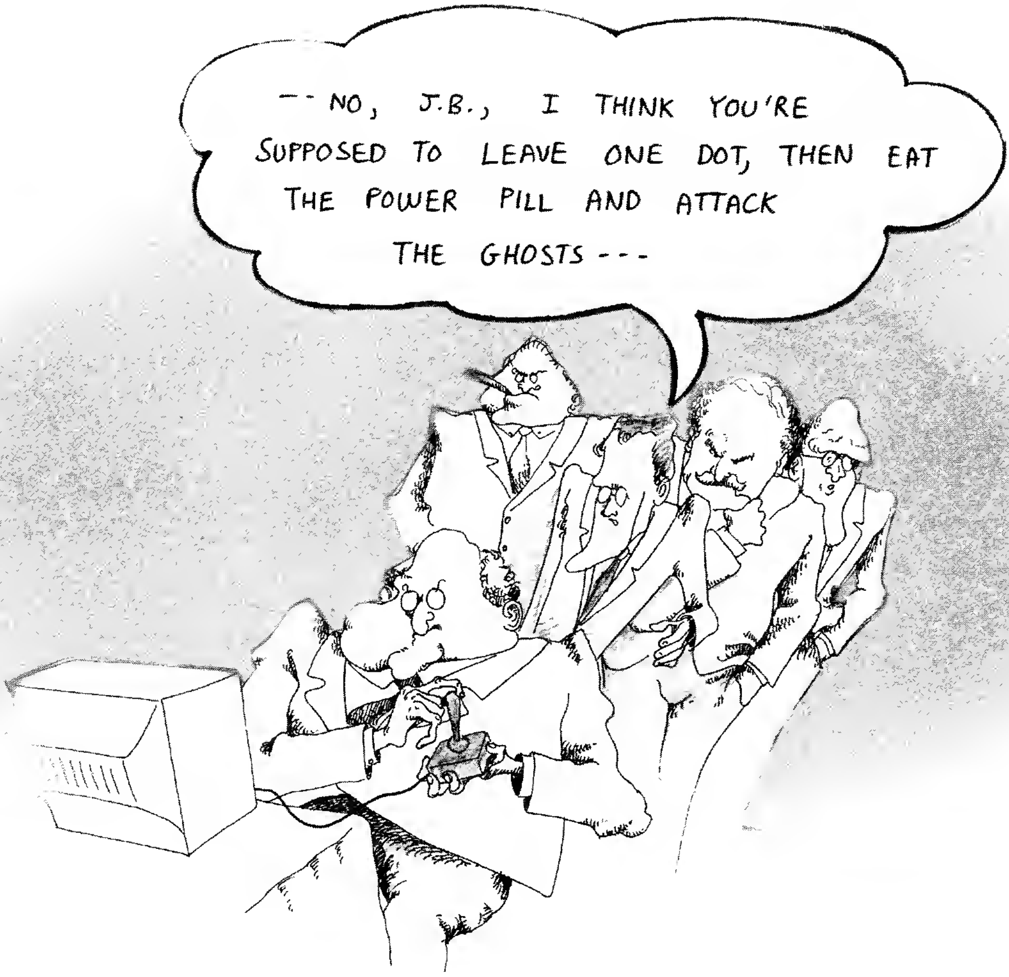
```
10020 IF PIC = 1 THEN GOSUB 10060 ELSE
        GOSUB 10080
10030 POKE O , B
10040 NEXT O
10050 RETURN
10060 B = A OR 64
10070 RETURN
10080 B = A ANO 191
10090 RETURN
```

The program that follows is one dealing with the advanced topic of machine language. This program will give you an audio response when any of the keys on the keyboard have been hit. The program will stay in memory until the machine is turned off, and you will still be able to use BASIC.

```
10 CLS
20 PRINT@ 11, "KEY SOUNO"
30 PRINT@ 64, "THIS PROGRAM WILL
    MAKE AN AUOIO ";
40 PRINT "TONE WHEN ANY OF THE KEYS ARE"
50 PRINT "HIT ON THE KEYBOARD"
60 I = PEEK(39) * 256 + PEEK(40) - 75
70 CLEAR 500 , I
B0 I = PEEK(39) * 256 + PEEK(40) + 1
90 FOR B = I TO I + 70
100 REAO D
110 POKE B , O
120 NEXT 8
130 EXEC I
```

```
140 POKE 157,18Ø : POKE 158,74 : NEW
150 REM ***
16Ø REM *** DATA FOR ***          *** KEY PRESS ***
17Ø REM ***
18Ø DATA 52, 63, 32, 3, 126, Ø, Ø, 19Ø, 1, 1Ø7,
    49, 14Ø, 248, 175, 164, 48, 141
19Ø REM ***
2ØØ REM ***
21Ø DATA Ø, 5, 191, 1, 1Ø7, 53, 63, 52, 63, 134,
    63, 183, 255, 35, 48, 141, Ø
22Ø REM ***
23Ø REM ***
24Ø DATA 33, 16, 142, Ø, 8, 23Ø, 128, 193, Ø, 39,
    19, 31, 152, 247, 255, 32, 18
25Ø REM ***
26Ø REM ***
27Ø DATA 18, 18, 92, 38, 247, 31, 137, 49, 63, 38,
    241, 32, 227, 53, 63, 32, 192, 69, 149, Ø
```

Again, here is another machine language (M/L) program. This program is just like having power windows in your new Cadillac. It's nice to have, but you can live without it. Have you ever typed in a command, such as CLOADM, turned away to check on something else, and come back to find nothing has happened but an ?SN ERROR? The following routine will give you a sort of "bouncing bonging" sound when you happen to get any type of error. This program will also stay in memory until the computer is turned off or until you type POKE 113,Ø and hit the RESET button on the back of the computer. (CAUTION: The POKE and hitting RESET will wipe out any BASIC program in the computer.) Both the


-- NO, J.B., I THINK YOU'RE SUPPOSED TO LEAVE ONE DOT, THEN EAT THE POWER PILL AND ATTACK THE GHOSTS ---

key sound and the error sound routines are completely compatible with each other. This means it is possible to have both programs in memory at the same time and have them operate properly together.

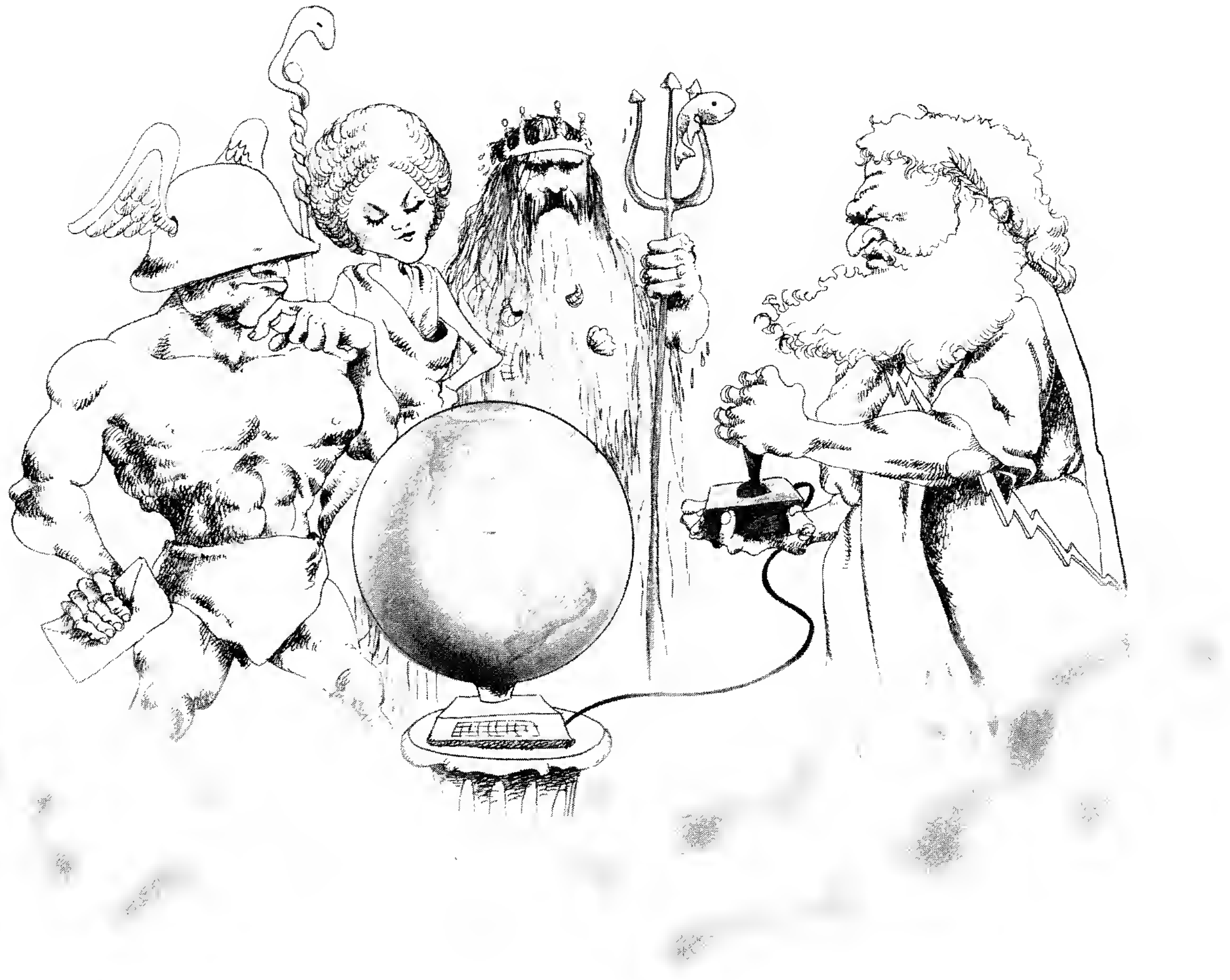
```
10 CLS
20 PRINT@ 1Ø, "ERROR SOUNO"
30 PRINT@ 64, "THIS PROGRAM WILL GIVE YOU AN"
4Ø PRINT "AUOIO DETECTION OF A SYNTAX"
5Ø PRINT "ERROR OR ANY OTHER TYPE OF ERROR"
6Ø I = PEEK(39) * 256 + PEEK(4Ø) - 11Ø
7Ø CLEAR 5ØØ , I
8Ø I = PEEK(39) * 256 + PEEK(4Ø) + 1
9Ø FOR B = I TO I + 1Ø4
1ØØ REAO O
11Ø POKE B , O
12Ø NEXT B
13Ø EXEC I
14Ø POKE 157,18Ø : POKE 158,74
15Ø PRINT
16Ø PRINT" . . . . . . . . . . . . OONE . . . . . . . . . . . . . . "
17Ø NEW
18Ø REM *** OATA FOR ***          *** ERROR ***
19Ø REM *** SOUNO ***
2ØØ REM ***          ***
21Ø DATA 52, 63, 32, 3, 126, 17, 148, 19Ø, 1, 143,
     49, 14Ø, 248, 175, 164, 48
22Ø OATA 141, Ø, 5, 191, 1, 143, 53, 63, 52, 63,
     134, 63, 183, 255, 35, 48
23Ø OATA 141, Ø, 33, 16, 142, Ø, 8, 23Ø, 128, 193, Ø,
     39, 19, 31, 152, 247
```

```
240 DATA 255, 32, 18, 18, 18, 92
250 DATA 38, 247, 31, 137, 49, 63, 38, 241, 32, 227,
     53, 63, 32, 192, 255, 16
260 DATA 229, 21, 224, 37, 208, 53, 192, 69, 208,
     85, 240, 32, 224, 48, 208
270 DATA 48, 192, 64, 176, 96, 160, 112, 144,
     128, 144, 160, 165
280 DATA 176, 181, 192, 197, 208, 224, 229, 0
```

This next program is one of those games that most everyone can enjoy. May they be 9 or 90, everyone likes Hang Man. The program can have up to 255 different messages up to 57 characters long. The program is long but well worth the effort of typing it in. You can have friends or family type in different messages and save them to Tape or Disk. Later you can come back and try to guess the messages left by them. Enter all messages between quotation marks in the A$( ) array, changing the ones in the listing if you like. For example, you might want to change A$(1) to equal "microcomputer" or add A$(200) to be "arithmetic". Well, have fun and don't get HUNG up.

```
10 CLEAR 500 : C = 1 : DIM A$(255)
20 A$(1) = "COLOR COMPUTER"
30 A$(2) = "COLOR HANG MAN"
40 A$(3) = "MAX CHARACTERS IN "
50 A$(4) = " ONE LINE"
60 A$(5) = "IS '57' AND YOU MAY"
70 A$(6) = "HAVE AS MANY MESSAGES"
80 A$(7) = "AS YOU WANT. 0-255"
90 A$(8) = " "
```

```
1ØØ A$(9) = " "
11Ø A$(1Ø) = " "
12Ø CLSØ
13Ø GOSUB 52Ø
14Ø A = LEN(A$(C))
15Ø USED = 416
16Ø SEN = 448
17Ø PRINT@ SEN, "SENT: ";
18Ø FOR B = 1 TO A
19Ø D$ = RIGHT$( A$(C) ,B )
2ØØ O = ASC( O$ )
21Ø IF D = 32 THEN PRINT@ SEN + 5 + A - B, " ";
    : T = T + 1 ELSE PRINT@ SEN + 5 + A - B, "-";
22Ø NEXT B
23Ø PRINT@ USEO, "USEO:";
24Ø PRINT@ 18Ø, "LETTER";
25Ø Z = Ø
26Ø INPUT AN$
27Ø IF AN$ = "" THEN GOTO 24Ø
28Ø FOR B = 1 TO A
29Ø D$ = RIGHT$( A$(C) ,B )
3ØØ O = ASC( O$ )
31Ø AN = ASC( AN$ )
32Ø IF AN = O THEN GOSUB 37Ø
33Ø NEXT B
34Ø IF Z < > 1 THEN 45Ø
35Ø PRINT@ 1B6, " ";
36Ø GOTO 24Ø
37Ø FOR F = 1 TO A
38Ø IF PEEK( 1Ø24 + SEN + 5 + A - B ) = AN THEN 24Ø
39Ø NEXT F
4ØØ PRINT@ SEN + 5 + A - B, CHR$( AN );
```



```
41Ø T=T+1
42Ø IF T = A THEN GOTO 157Ø
43Ø Z = 1
44Ø RETURN
45Ø PRINT@ USED + 5, CHR$( AN );
46Ø USED = USED + 1
47Ø ON USED-416 GOSUB 6ØØ,74Ø,82Ø,87Ø,92Ø,
    97Ø,1Ø2Ø,1Ø7Ø,112Ø,117Ø,12ØØ,
    127Ø,132Ø,135Ø
48Ø GOTO 24Ø
49Ø REM ***
5ØØ REM *** POLE ***
51Ø REM ***
52Ø FOR Y = 21 TO Ø STEP -1 : SET(2,Y,7) : NEXT Y
53Ø FOR X = 3 TO 18 : SET(X,Ø,7) : NEXT X
54Ø POKE 512, C
55Ø RETURN
56Ø REM ***
57Ø REM *** HEAD ***
58Ø REM ***
59Ø SOUND 56,5
6ØØ SET(18,1,5)
61Ø FOR X = 16 TO 2Ø : SET(X,2,4) : NEXT X
62Ø FOR X = 15 TO 21 : SET(X,3,4) : NEXT X
63Ø SET(14,4,5) : SET(22,4,5)
64ØSET(13,5,5) : SET(14,5,5) : SET(22,5,5) : SET(23,5,5)
65ØSET(12,6,5) : SET(14,6,5) : SET(22,6,5) : SET(24,6,5)
66Ø SET(13,7,5):SET(14,7,5):SET(22,7,5):SET(23,7,5)
67Ø SET(14,8,5) : SET(22,8,5)
68Ø SET(15,9,5) : SET(21,9,5)
69Ø FOR X = 16 TO 2Ø : SET(X,1Ø,5) : NEXT X
7ØØ RETURN
```

```
710 REM ***
720 REM *** BODY ***
730 REM ***
740 SET(18,11,6)
750 FOR X = 15 TO 21 : SET(X,12,6) : NEXT X
760 FOR Y=13 TO 17 : SET(14,Y,6) : SET(22,Y,6) : NEXT Y
770 FOR X = 14 TO 22 : SET(X,17,6) : NEXT X
780 RETURN
790 REM ***
800 REM *** RIGHT ARM ***
810 REM ***
820 SET(13,13,5) : SET(12,12,5) : SET(11,11,5)
830 RETURN
840 REM ***
850 REM *** LEFT ARM ***
860 REM ***
870 SET(23,13,5) : SET(24,14,5) : SET(25,15,5)
    : SET(26,14,5)
880 RETURN
890 REM ***
900 REM *** RIGHT LEG ***
910 REM ***
920 FOR Y = 18 TO 19 : SET(17,Y,5) : NEXT Y
930 RETURN
940 REM ***
950 REM *** LEFT LEG ***
960 REM ***
970 FOR Y = 18 TO 19 : SET(19,Y,5) : NEXT Y
980 RETURN
990 REM ***
1000 REM *** RIGHT HAND ***
1010 REM ***
```

```
1020 SET(10,10,4) : SET(9,11,4)
      : SET(11,9,4) : SET(9,9,4)
1030 RETURN
1040 REM ***
1050 REM *** LEFT HAND ***
1060 REM ***
1070 SET(27,13,4) : SET(28,14,4) : SET(26,12,4)
      : SET(28,12,4)
1080 RETURN
1090 REM ***
1100 REM *** RIGHT FOOT ***
1110 REM ***
1120 SET(15,19,5) : SET(16,19,5)
1130 RETURN
1140 REM ***
1150 REM *** LEFT FOOT ***
1160 REM ***
1170 SET(20,19,5) : SET(21,19,5)
1180 RETURN
1190 REM ***
1200 REM *** RIGHT EYE ***
1210 REM ***
1220 SET(16,5,3)
1230 RETURN
1240 REM ***
1250 REM *** LEFT EYE ***
1260 REM ***
1270 SET(20,5,3)
1280 RETURN
1290 REM ***
1300 REM *** NOSE ***
1310 REM ***
```

```
1320 SET(18,6,8)
1330 RETURN
1340 REM ***
1350 REM *** MOUTH ***
1360 REM ***
1370 SET(17,8,4) : SET(18,8,4) : SET(19,8,4)
1380 SOUND 78, 8
1390 FOR A = 1 TO 70 : NEXT A
1400 SOUND 78, 8
1410 FOR A = 1 TO 70 : NEXT A
1420 SOUND 78, 3
1430 SOUND 78, 5
1440 SOUND 108, 8
1450 SOUND 99, 4
1460 SOUND 99, 4
1470 SOUND 78, 5
1480 SOUND 78, 5
1490 SOUND 69, 5
1500 SOUND 78, 5
1510 FOR A = 1 TO 463 * 3 : NEXT A
1520 CLS
1530 PRINT "YOU'RE HUNG!"
1540 PRINT : PRINT
1550 PRINT "THE ANSWER IS: ";A$(C)
1560 GOTO 1610
1570 FOR A = 1 TO 436 * 6 : NEXT A
1580 CLS
1590 PRINT
1600 PRINT "GOOD WORK!!!"
1610 PRINT@ 128, "DO YOU WANT
     TO TRY ANOTHER";
1620 INPUT A$
```

```
1630 IF A$ = "YES" DR A$ = "Y" THEN CLEAR
       : C = PEEK(512) + 1 : PDKE 512, C : GDTO 20
1640 PRINT : PRINT
1650 PRINT "DK. GOOD BYE."
1660 END
```

This last program will help you quickly look up the names and addresses of your friends. All you have to do is put their names and addresses into the DATA statements beginning on line 200 as the example shows. Be sure to have DATA for all five string variables (NAMES$, AD$, CITY$, S$, and ZIP$). If you ask it for a name it cannot find, the program will eventually find END in the DATA statement in line 1000. At that point the program will quit. So be sure to spell the names correctly when you RUN the program. (PSST... Do you want to add the phone number? Just add line 85 READ PH$ and add the phone number to the DATA statements. You figure out the rest.)

```
10 REM ***********************
20 REM NAME AND ADDRESS
30 REM ***********************
40 CLS
50 INPUT "ENTER NAME TD FIND ";N$
60 READ NAME$ : IF NAME$ = "END" THEN END
70 READ AD$ : READ CITY$
80 READ S$ : READ ZIP$
90 IF NAME$ = N$ THEN 100 ELSE 60
100 REM ****************************
110 REM PRINT THE INFDRMATIDN
120 REM ****************************
```

```
130 CLS
140 PRINT NAME$
150 PRINT AD$
160 PRINT CITY$;", ";S$;" ";ZIP$
170 FOR V = 1 TO 4 : PRINT : NEXT
180 INPUT "ANOTHER(Y/N) "; AN$
190 IF AN $ = "Y" THEN 5Ø
2ØØ REM *************************
21Ø REM RECORD INFORMATION
22Ø REM *************************
23Ø OATA KATHY SMARTS, 68Ø9 DIGITAL CT.,
     SILICONE, CA, 92129
24Ø OATA KARL COMPUTE, 1Ø24 MICRO BLVD,
     OPCODE, IOWA, 12345
25Ø REM **************************
26Ø REM JUST KEEP ON AOOING
27Ø REM NAMES AND ADDRESSES
28Ø REM BETWEEN HERE AND
29Ø REM LINE 1ØØØ.
3ØØ REM **************************
1ØØØ DATA END
```

That ought to give you plenty of programs you can use. See if you can invent some of your own programs. In the next chapter, we'll tell you how you can have even more statements to help you program with Extended BASIC.

15

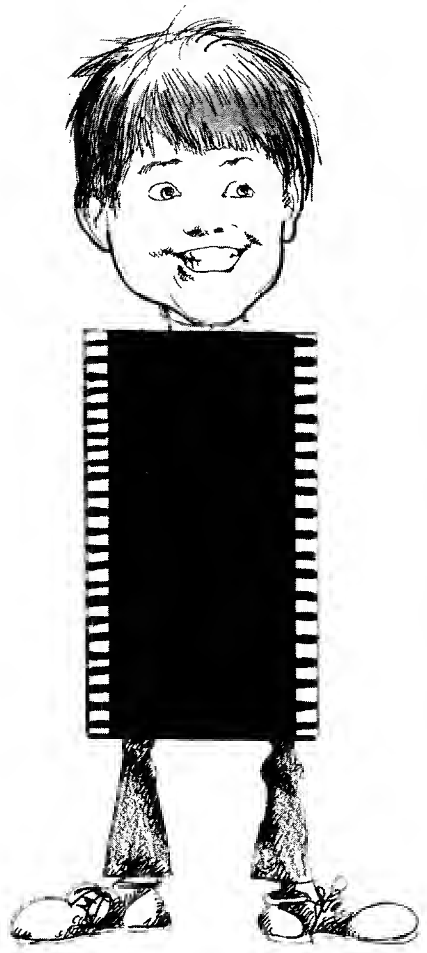
EXTENDED COLOR BASIC

By now you should know that your Color Computer is not a toy nor an arcade game with a keyboard. If you have just regular Color BASIC, you have a very powerful computer; but if you have Extended Color BASIC, you have a super powerful computer with some advanced features. Most people who buy the Color Computer start out with Color BASIC first. Then they realize that it sure would be nice to tap into the superb graphics and the great sounds that can be produced by having the Extended chip. Extended BASIC may be purchased from your Radio Shack computer center, or there are certain places that sell them in a kit. The cost is between $99 - $12Ø. Radio Shack includes the warranty installation. (If you know something about hardware, though, the Extended BASIC chip is easy to install.)

Here we'll discuss the major abilities of Extended BASIC. If you want a complete list of all the Extended keywords (statements, functions, etc.), use the Red fold-out card that came with your computer.

EDITING WITH EXTENDED BASIC

Have you ever typed in a line and found out that you had an error or misspelled a word? Well, we can say we definitely have. Normally you would have to retype the



whole line over again, but not if you have the EDIT function that's in Extended BASIC. The EDIT function will allow insertion of new characters, deletion of unwanted characters, and searching for characters. For example, suppose you had the following line:

```
4Ø FOR X 1 TO 5Ø : PRINT X : NEXT X
```

You forgot to put in the = (equal sign) between the first X and the 1. To change that with the EDIT function, you would enter:

```
EDIT 4Ø <ENTER>
```

The line will appear on your screen, and as you press the space bar the line will begin to appear. When you get to the X, press the "I" key (for insert) and enter the equal sign. Press ENTER, and your line will be edited. That's a lot simpler than re-writing the entire line.

There are a few other commands of the editor, but we wanted you to see how to use EDIT specifically. That particular function alone is well worth the cost of Extended BASIC.

Would you like to show off your computer or make some spectacular graphics? Well with the CIRCLE, COLOR, DRAW, GET, LINE, PAINT, PCLEAR, PCLS, PCOPY, PMODE, PPOINT, PRESET, PUT, and SCREEN functions, you can produce some of the most fascinating graphics imaginable. For example, with the statement CIRCLE you can draw a circle. The following program is an example of how simple it is:

```
10 PMODE 1,1
20 PCLS
30 SCREEN 1,1
40 CIRCLE (100,100),50
50 GOTO 50
```

The CIRCLE requires only that you specify the center in terms of X,Y coordinates (1ØØ,1ØØ in our example), and its radius, 5Ø, in this case. You can include other parameters as well to make different colors, beginning and end as well as its height /width to make ovals. The other commands give you equal flexibility in working with computer graphics.

Do you want to compose music? Here's your chance to do so with Extended's PLAY command. It has the ability to play five different octaves, flats, sharps and variable tempos. These are commands available with PLAY. It can play music with notes that are specified as A-G or 1-12, (V) for 32 levels of volume, (T) how fast the tempo is, (O) 1-5 different sound octaves, (L) for varied vote length, (X) allows execution of a substring, and (P) pause. Also, you have the ability to produce flats and sharps with the notations of (+ or #) for the sharps and (–) for the flats.

Extended BASIC also has the ability to do complex mathematical calculations not found in the Standard Color BASIC. This pretty well summarizes what Extended BASIC can do, and to our feelings no serious Color Computer owner should be without it.

YOU MUST RECTIFY YOUR ERRORS!
THIS'D BE EASIER WITH EXTENDED BASIC...

TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KID TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KI

* * * GLOSSARY * * *

ARRAY An arrangement or pattern of data in succession, such as a table of numbers. Also a sequentially stored set of variables.

ASCII An acronym for American Standard Code. It is a standard code adopted by most computers to remain compatible with others.

BASIC An acronym for Beginner's All-purpose Symbolic Instruction Code. This language is easy to learn and easy to use.

BINARY Pertains to the number system with only two digits, either 1 or Ø.

BOOLEAN ALGEBRA A portion of logic which is similar to algebra, but deals with logical relationships not numeric ones.

BRANCH Alternative directions a program can take.

BUG A mistake in a computer program, or a malfunction in the computer's hardware.

CASSETTE A means of magnetic tape storage for programs, data, and files.

CELL A storage place in the part of memory with the capacity to hold only one bit.

CENTRAL PROCESSING UNIT The main component of any computer, the brain of the operations. The CPU on the Color Computer is a 68∅9 microprocessor.

CHARACTER Any letter, number, or symbol stored by the computer.

COBOL An acronym for COmmon Business Oriented Language. A high level language for the development of complex business programs.

COMPUTER A device that executes mathematical or logical operation without the help of humans.

COMPUTER OPERATOR A person who knows the programming language and is able to operate peripheral devices.

COMPUTER UTILITY A program or device that allows the user to gain a better use of the computer's functions.

CRT An acronym for Cathode Ray Tube: The computer's T.V. or monitor is the CRT.

DATA The software of the computer; also a vital part of the computer's operation. Also a statement in BASIC.

DEBUG To find and fix a problem that might exist in a program.

DECIMAL DIGIT A numbering system that is base 1∅ and whose numbers include ∅, 1, 2, 3, 4, 5, 7, 8, 9.

DELETE To completely eliminate or remove.

DIGITAL COMPTER A device that operates on a base of ones and zeros.

DISK DRIVE A form of storage for programs, files, data using hard or floppy disks or diskettes.

DISKETTE The media for the disk drive.

DOS An acronym for Disk Operating System.

EDIT To check, change, or insert data into a program.

ERROR A problem or bug in a program usually caused by the operator of the computer.

FILES Programs or data that are stored on tape or diskette and can be called up again for later use.

FIRMWARE Software that is permanently stored in the computer. Storage media is ROM (Read Only Memory).

FLOWCHART A diagram that shows the logical operation of a program with the use of various symbols.

FORTH A high level fast language that uses user defined words to create programs. This language is available for the Color Computer.

FORTRAN An acronym for FORmula TRANslator. A high level language used primarily for making highly complex scientific and engineering computations.

GLITCH A sudden jump in the AC line voltage or other source voltages that may cause unexpected wipe-outs of computer memory.

GRAPHICS A mode that allows the computer to form colorful or complex visual displays and drawings.

HARDCOPY A printed copy of the output of a program, listing, or graphic display.

HARDWARE The physical part of the computer. Keyboard, CPU, RAM chips, ROM chips, etc.

HEXADECIMAL A numbering system that is on the base of 16 and whose digits include Ø, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.

HOME COMPUTER A small low-cost computer that is generally from $1ØØ.ØØ to $1ØØØ.ØØ dollars. We think they were invented so that kids could have a computer in their homes.

INPUT Inserting information into the computer.

INPUT/OUTPUT Means of sending and receiving information such a Disk Drive or Cassette Deck. Usually abbreviated as I/O.

INFORMATION The flow of data from one point to another.

INTEGRATED CIRCUIT A complex complete circuit that requires minimal parts to get a desired circuit operation such as a computer.

I/O An abbreviation for Input Output.

K An abbreviation for kilo or 1ØØØ. (Actually refers to 1Ø24.)

KEYBOARD A group of keys manually operated for inputting information into the computer.

KEYWORD A major word element in a programming language. In BASIC, such words as RUN, FOR, NEXT, GOTO, PRINT are keywords.

LIGHT PEN An electronic device that allows input to the computer by the use of light or darkness on the CRT screen.

LOAD To read in information from an external device such as a Disk Drive and/or Cassette Deck.

MEMORY The part of the computer that allows storage of programs and data.

MICROCOMPUTER A small, low-cost computer that is generally used for home, small business use, and is especially made for kids.

PASCAL A highly structured programming language originally used to help students learn structured programming.

PROGRAM A set of instructions that guides and informs the computer what exactly the user wants to do.

RAM An acronym for Random Access Memory. The process of obtaining data from or placing data into storage (memory).

RUN A single statement for executing a program.

SAVE To save information to an external device such as a Disk Drive and/or Cassette Deck.

STRING A variable that stores any kind of characters generated by the computer in ASCII form.

SYNTAX The grammatical and base structural laws of a language.

WHOLE NUMBER A number without any fractional parts; e.g., 12, 45, 54, 99 , not 1-2/3, 4/5. Also referred to as an INTEGER.

WORD A complete word is comprised of eight bits.

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KII

FOR THE RADIO SHACK COLOR COMPUTER

Written by kids for kids, this unique book explains BASIC programming on the Radio Shack Color Computer. Created from the idea that kids can teach other kids better than anyone else, the material is designed to help you get started using and programming your Color Computer.

You'll learn how to use the cassette and disk drive, PRINT and math statements, variables, loops, branching & subroutines, and arrays. Two chapters are devoted to sound and graphics and another will teach you how to write an original game. Before long, you'll be using your Color Computer to finish your homework in record time!

As an added bonus, the authors discuss their favorite programs and games. Complete with a computer glossary and index, KIDS TO KIDS ON THE COLOR COMPUTER will teach you the magic of programming in simple, straightforward language.

COMING SOON FOR THE APPLE II, II+ & //e and THE COMMODORE 64

DATAMOST™ INC
8943 Fullbright Avenue, Chatsworth, CA 91311-2750
(818) 709-1202

ISBN 0-88190-231-4

0 48831 00017